University of Tartu
Department of Computer Science
Norwegian University of Science and Technology
Department of Telematics

Mohammad Jabed Morshed Chowdhury

Modeling Security Risks at the System Design Stage

Alignment of Mal Activity Diagrams and SecureUML to the ISSRM Domain Model

Master's Thesis (30 EAP)

Supervisor: Dr. Raimundas Matulevičius, University of Tartu.
Prof. Guttorm Sindre, IDI, Norwegian University of Science and Technology.
Dr. Peter Karpati, IDI, Norwegian University of Science and Technology.

Author _____, _____, June, 2011

Supervisor _____, _____, June, 2011

Approval for Defense

Prof.'s Name _____, _____, June, 2011

TARTU, 2011

Abstract

Security engineering is one of the important concerns during system development. It should be addressed throughout the whole system development process; however in many cases it is often dealt only during system development and maintenance.

There are several security modeling languages (e.g, Misuse case, Secure Tropos) that help dealing with security risk management at the requirements stage. In this thesis, we are focusing on the modeling languages (e.g. Mal activity diagrams and SecureUML) that are used to design the system. More specifically we investigate how these languages support information systems security risks management (ISSRM).

The outcome of this work is an alignment table between the Mal activity diagrams and SecureUML language constructs to the ISSRM domain model concepts. We ground our analysis and validate the received results on the number of illustrative examples. We hope that our results will help developers to understand how they can consider security risks at the system design stage. In addition we open the way for the interoperability between different modeling languages that are analysed using the same conceptual background, thus, potentially leading to the transformation between these modeling approaches.

Acknowledgements

At this very moment of my Master's thesis submission, I would like to convey my sheer gratitude to the Almighty and some special people for their insightful help and care.

I am deeply grateful to my supervisor, Dr. Raimundas Matulevičius, in University of Tartu, for his detailed and constructive comments and important support throughout this work. With his enthusiasm, inspiration and great efforts to explain things clearly and simply, he helped me a lot to approach, analyze and solve the research problem systematically. Throughout my thesis writing period, he provided encouragement, sound advice, good teaching, good company and lots of good ideas. I would have been lost without him.

I would like to express my deep and sincere thanks to my co-supervisor Professor Guttorm Sindre and Dr. Peter Karpati in Norwegian University of Science and Technology (NTNU) for their time, discussions and valuable comments.

I also want to thank all my friends both in Norway and Estonia for their continuous love, affection and care during my stay away from my own country. Last but not the least, I wish to thank my parents who are always been my guide and the source of inspiration.

Tartu, 14th June, 2011

Mohammad Jabed Morshed Chowdhury

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

C&A	Certification and Accreditation
CERT/CC	Computer Emergency Response Team/ Coordination Center
CORAS	Risk Assessment of Security Critical Systems
DITSCAP	Department of Defense Information Technology Security Certification and Accreditation Process
DoD	Department of Defense
IS	Information System
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
ISSRM	Information System Security Risk Management
KAOS	Knowledge Acquisition in automated Specification
KeS	KOAS with Security Extenstion
OCL	Object Constraint Language
RBAC	Role Based Access Control
RE	Requirement Engineering
RM	Risk Management
ROSI	Return On Security Investment
SQUARE	Security Quality Requirements Engineering
UML	Uniform Modeling Language

CHAPTER 1

Introduction

Nowadays, Information Systems (IS) have large impact on our social and economical life. Facebook, Twitter and e-Commerce sites are common phenomena in our daily life. Information system also has become an integral part of organizational structure and commerce. Introduction of information system has made our life easy, convenient and more relaxing. But like many other opportunities information system also has its drawbacks. Hacking of websites and stealing credit card information are not rare in Internet world. To undermine all kind of potential attacks, information system needs to be secured. The significance of security technologies is now widely accepted and is therefore receiving a continuous attention. As a result security facets play a vital role in any modern information system.

But security mechanism is not free, it needs investment. So, Return On Security Investment (ROSI) has become a major concern (Mayer, 2009) in many organizations. This involves a risk management processes to justify investment for security measures and the business assets to be protected. This creates new approach in organization called security risk management. To support security risk management, security mechanism should be addressed and realized at all the stages of the information system development.

1.1 Scope

This research work is about the design stage support for security risk management. In this work, we will mainly focus on security risk management for information system. In the following section we will introduce different related concepts and boundaries of this work.

1.1.1 Information System (IS)

The term *information system* is defined in different ways related to the domain it is used. For our work we choose the definition of information system like this, "*A system for dissemination of data between persons - potentially, to increase their knowledge"* (Turban *et al.*, 2010). It is clear from the definition that we will include both information technology (*A system for dissemination of data*) and people's (*between persons*) activities in this work as information system. Our focus will be at the design stage of the information system.

1.1.2 Security Engineering

Security is usually seen in two different ways, dedicated malicious act and/or accidental harm to the system or to the organization. Here we take the definition as "*security is the degree to which malicious harm is prevented, detected, and reacted* ." (Firesmith, 2003). Security covers broad range of areas including financial, environmental, information system. Here, we will only work with the security concepts at design stage of the information system.

1.1.3 Risk Management

Risk is a very general concept and applies to different domain. Risk can be seen as combination of the probability of an event and its negative consequence. Risk management (RM) is defined as "*coordinated activities to direct and control an organization with regard to risk*" (ISO, 2002). Risk management can be related to finance, organization, environment and security etc. In this work we will only focus on the aspects of security risk management for information system at the design stage.

So the overall scope of this work will be information system security risk management at the design stage (Figure 1.1).

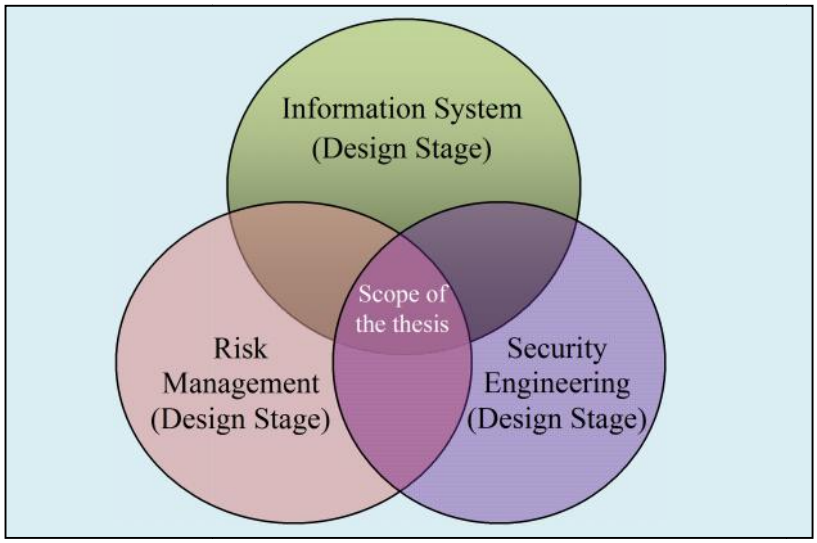

Figure 1.1: The Scope of this Thesis Work

1.2 Motivation

Though *security* becomes a buzz word for information technology world but it is often seen as ad-hoc basis during the later part of the development cycle (Sindre, 2007). This approach (ad-hoc) often leads to errors and vulnerabilities that provides a potential for exploitation. Statistics from CERT/CC[1], shows that the number of security incidents have increased quite significantly from 2000-2007 (Figure. 1.2).

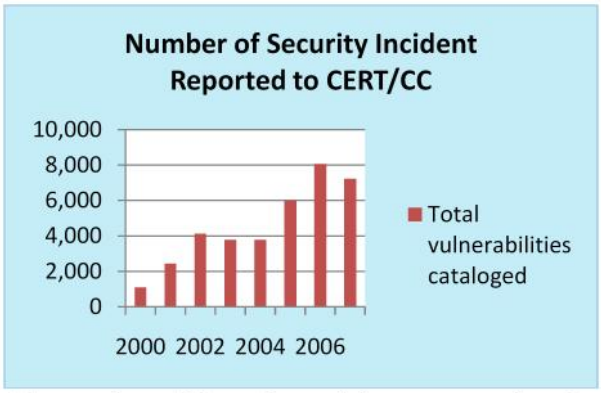

Figure 1.2: Number of Security Incident Reported to CERT/CC

[1] http://www.cert.org/stats/

A large number of software vulnerabilities have indicated that these systems are repeatedly lacking the appropriate measures from design and implementation perspective. Any minor flaw at the design stage can contribute significantly to the application vulnerabilities, which can be exploited by the attackers. A study (Hoglund and McGraw, 2004) shows that design stage problems were accounted for about 50% of the security flaws during the Microsoft's "security push" in 2002. Viega and McGraw have mentioned design stage vulnerabilities as the second major cause of security risk in software in their book (Viega and McGraw, 2002).

By following an improved and well structured process at design stage, with security in mind can reduce these vulnerabilities quite significantly. Thus this work is motivated to help developers to understand how they can consider security risks at the system design stage.

1.3 Mission Statement

In this section, the research problem and research questions are explained. Our research work is concentrated on design stage support for analyzing security risk management. Thus, the research problem of this work is,

Research problem: *Support security modeling language for security risk management at the design level.*

This work targets to show how the design stage modeling language supports security risk management. To do so, we will first explore security risk management methods and frameworks to understand how security risks are managed. Then we will investigate different modeling languages and particularly design stage modeling languages. We will look into their concepts (e.g., syntax and semantics) and will align them with the security risk management concept.

The work does not develop any design level modeling language, because of the limited resources and time available. Instead, the work discusses about the concept alignment (we do not mean the exact match but rather correspondence, similarity or overlap of concepts) of the exiting design level modeling language to help developers to address security risk at design stage. To understand and deal with the research problem, it is divided into three research questions.

RQ 1: *Is there any domain model which would help to understand security risk management at design stage?*

This question will investigate the available security methods (e.g., Braber *et al.,* 2007, Mead *et al.* 2005) and security risk management frameworks (e.g., Haley *et al.* 2006, Lee *et al.* 2007). This will help us to understand the concepts of security risk management and how they are managed. The outcome of this question will be a selection of security risk management domain model which we will use to analyze the modeling language at the design stage.

RQ 2: *What are potential security modeling languages at the design stage to support security risk management?*

This question will look into different modeling languages, specially design stage modeling languages. In this work, we will focus on UML based approaches which have security extension (e.g., Misuse cases (Sindre and Opdahl, 2001), Abuse cases (McDermott and

Fox, 1999) etc). However, to understand how the modeling languages could be considered with respect to security risk management, we will also look into two goal-oriented languages, *i.e.,* Secure Tropos (Mouratidis, 2005) and KAOS with security extension (Lamsweerde, 2004). We will mainly work with the concepts of design stage modeling language (e.g., Mal activity diagrams (Sindre, 2007) and SecureUML(Lodderstedt *et al.,* 2002)).

RQ 3: *How could the security modeling language support security risk management at the design stage?*

The question involves the analysis of concepts of the design stage modeling languages (Mal activity diagrams (Sindre, 2007) and SecureUML(Lodderstedt *et al.,* 2002)) with respect to security risk management. We will work to find the concept alignment between design stage modeling language and security risk management for information system. The outcome of this question will be concept alignment tables. This alignment table will help software developer to better understand security risk at the design phase.

The working hypothesis of this study is that if the developers understand and can realize security risk at the design stage, it will help them to develop secured system. Last but not the least, it needs to be clearly mentioned that this understanding will help to better deal with security vulnerabilities at design stage but will not alone solve all the problem.

1.4 Thesis Outline

The thesis is organized in five parts and eight chapters.

Part I, Background Study contains three chapters. The second chapter is related to the theoretical overview of the current practices and research related to security risk management approaches. The third chapter discusses about existing concept alignment between requirement engineering modeling languages and information system security risk management (ISSRM) (Dubois, 2010) concepts. The fourth chapter investigates the security extension of UML diagrams, specially for design.

Part II, Security Risk Management for IS Design contains two chapters. The fifth chapter discusses about the concept alignment between Mal activity diagrams and ISSRM domain model. The sixth chapter shows the concept alignment between SecureUML and ISSRM domain model. Both of this alignment is done by using a running example described in Annex A.1.

Part III Validation contains one chapter. This chapter discusses about a case study (Section 7.2) which investigates the validity of our proposal in Table 5.1 and Table 6.1.

Part IV, Conclusions and Future Work summarizes the major findings and discusses the future work. It also draws the conclusion and highlights the claimed contribution.

PART I

Background Study

The purpose of this part is to survey the state-of-the-art of the security risk management approaches, existing alignment between requirement engineering language and ISSRM domain model and modeling languages at design stage. The part consists of three chapters.

The second chapter *Security Risk Management of Information System* reviews the security methods and security risk management focusing on information system. Overview of the security method and security risk management include three approaches each. The objective of this chapter is to choose a security risk management domain model for information system which we will use in later chapter. This model will be used to investigate how the design stage modeling language supports the security risk management of information system. In the process of selecting the domain model, we investigate three security method: Security Quality Requirements Engineering (SQUARE) Methodology (Mead *et al.* 2005), Framework by Haley (Haley *et al.,* 2008) and Framework by Lee (Lee *et al.,* 2005) and three security risk management approaches: CORAS (A Platform for Risk Analysis of Security Critical Systems) (Braber *et al.,* 2007), Tropos Goal Risk Model (Asnar *et al.,* 2007) and Information System Security Risk Management (ISSRM) Domain Model (Dubois *et al.*, 2010). After discussing concepts of these security methods and security risk management methods, ISSRM domain model is selected.

In the third chapter *Security Risk Management for IS Requirement Engineering,* we study concept alignment of Secure Tropos (Mouratidis, 2005), KOAS with Security Extension (Lamsweerde, 2004) and Misuse case (Sindre and Opdahl, 2001) with ISSRM domain model. This chapter contributes with understanding of the research method that we apply to align the design stage languages with ISSRM domain model. In first section of the chapter we analyze the research method that has been used in these alignments. Later in the chapter, we present a concept alignment table which contains the ISSRM domain model concepts and their alignment with three requirement engineering modeling languages.

The fourth chapter *Security Modeling Language for System Design* surveys different modeling languages. The purpose of this survey is to understand design stage modeling language concepts. We investigate Abuse cases (McDermott and Fox, 1999), Misuse cases (Sindre and Opdahl, 2001), Mal activity diagrams (Sindre, 2007), UMLSec (Jurjens, 2002) and SecureUML (Lodderstedt *et al.,* 2002). Among these languages Abuse cases and Misuse

cases are used at the requirement engineering and Mal activity diagrams, UMLSec and SecureUML are used at the design stage. We investigate the concrete syntax, meta model, semantics and their processes by using an example presented in Annex A.1. The outcome of this chapter is that we select two design stage modeling languages: Mal activity diagrams and SecureUML.

CHAPTER 2

Security Risk Management of Information System

Various security frameworks (e.g., Haley *et al.* 2006, Lee *et al.* 2007) and security risk management methods (e.g., Braber *et al.,* 2007, Mead *et al.* 2005) have been proposed to investigate, analyze and risk treatment for security risk management. We will discuss some methods and frameworks here to understand the key concepts regarding security risk management for information system. We have selected SQUARE (Security Quality Requirements Engineering) method (Mead *et al.,* 2005), CORAS (A Platform for Risk Analysis of Security Critical Systems) (Braber *et al.,* 2007), a framework proposed by Haley *et al.* (Haley *et al.* 2006), ISSRM domain model (Dubois *et al.*, 2010) and a framework by Lee *et at.* (Lee *et al.* 2007) to understand the security risk management concepts at design stage. Based on this investigation we will answer our first research question: *Is there any domain model which would help to understand security risk management at design stage?* At the end of this chapter, we will summaries different features of these approaches to select a domain model for the next chapters (Chapter 5 and Chapter 6).

2.1 Security Methods

Today, information security is often conceptualized as being the protection or preservation of three key aspects of information: *availability, integrity* and *confidentiality*. Security methods are used to secure the data and other components of the information system by ensuring these three major concepts. In addition to the technology, security method also enquires the participation of everyone in the organization. In short, security methods are the procedures and guidelines that define how security should be implemented.

2. 1.1 Security Quality Requirements Engineering (SQUARE) Methodology

In requirement engineering, much attention is given on the functionality of the application and most of the time the security features are neglected in the early phase of the development process. When security features are added later in the system, they generally increase the cost of the system. A study (Hoo *et at.,* 2001) has showed that return on investment rises up to 12%-21% when security analysis and secure engineering practice are introduced in the early development cycle. Requirement engineering also suffers from few common problems. For example, all the relevant stakeholders are not taken into consideration and requirement analysis process in use cannot elicit all the requirements of the system owner. Sometimes requirements are not defined in a way that can be validated by the stakeholders or the system owner later on. To overcome this problem, SQUARE methodology is proposed (Mead *et al.* 2005). SQUARE is a nine steps process, which elicits, categorizes and priorities security requirements.

How to Make Use of SQUARE :
Security requirements are specified before the critical architectural and design decision to get the most benefit out of this methodology. In this way the critical business risk can be addressed at the requirement engineering stage. The nine steps of the SQUARE method are discussed below.

Step 1: Agree on Definitions. This is arranged as a focus group discussion where all the stakeholders agree on a set of terminologies and their definitions to be used in this security requirement activities.

Step 2: Identify Security Goals. In this step, the security requirements are aligned with the business policies. The security requirements of all kind of stakeholders are documented and addressed in this step. When goals of all the stakeholders are defined, they are then prioritized.

Step 3: Develop Artifacts to Support Security Requirements Definition. In this step, the necessary security artifacts are collected and created to support all the subsequent activities. Misuse cases, Abuse cases and Attack tree etc. are used to support requirement definition.

Step 4: Perform Risk Assessment. In this step, the security expert, requirement engineers and stakeholders participate in a discussion to select the appropriate risk assessment method for the organization. The artifacts from Step 3 are used as the input for this step. The risks are assessed by indentifying threat and vulnerabilities of the system and it can help to discover the high priority security exposure.

Step 5: Select Elicitation Techniques. The elicitation technique is selected for the security requirements based on the expertise of the stakeholders. A formal elicitation technique is more effective when the participants are diverse and with different expertise level. Elicitation can be structured interviews, survey or even by sitting down with the primary stakeholders.

Step 6: Elicit Security Requirements. This is the main process where the elicitation of the security requirement is done using the selected elicitation technique selected in the previous step. This is built by using the artifacts created in Step 3, such as misuse cases, abuse cases, attack trees and threat scenarios.

Step 7: Categorize Requirements. Requirement engineers categorizes the elicited security requirements based on the standard methods. This step also helps to priorities the activities in the next step.

Step 8: Prioritize Requirements. Prioritization of the elicited security requirements is done not only by the security techniques but also on the basis of the cost/benefit analysis in order to select the highest payoff security requirements.

Step 9 : Requirements Inspection. Security requirement inspection is done to provide the organization an initial set of prioritized security requirements. This also indicates the incomplete part which should be revisited at a later time.

In short, SQUARE is a procedure for eliciting, categorizing and prioritizing security requirements for information technology systems and applications. SQUARE indentifies security requirements by discussing with the stakeholders of the system and security experts.

This approach is focused on security requirements of information system. Though it is focused on requirement engineering stage but it gives us the understanding of security risk management for information system which we can use at design stage.

2.1.2 A Framework for Representation and Analysis of Security Requirements Engineering by Haley *et al.*

This framework (Haley *et al.,* 2008) is proposed by Haley *et al.* to define adequate security requirements to fulfill the security goals. This is based on three major concepts:

1. ***Definition*:** A clear definition of security requirements and related concepts.
2. ***Context*:** A more comprehensive guideline about the boundary of the system. All the assumptions and preconditions must be well defined for information system and its environment.
3. ***Satisfaction*:** There must be a mechanism to prove that whether any specific security requirement satisfies the security goal or not. It also checks whether information system satisfies the security requirements or not.

The security requirement framework described in this framework ensures the elicitation, validation and verification of security requirements and other artifacts by combining two major concepts, requirement engineering and security engineering (Firesmith, 2003). Functional goals, which constitute the functional requirement are taken from requirement engineering. From security engineering, it takes the concept of *asset, treats* and *harm*.

The main steps of this framework are described as,

1. Security goals are set to protect the *asset* from any kind of *harm*.
2. The security goals are realized into the security requirements. The security requirements set some constraints on the functional requirements to protect the *asset* from *harm*.
3. Sometimes security requirement may lead to additional (secondary) security requirements to countermeasure *threat*. Secondary security goals may call for detective or preventative measures of the system.
4. Security satisfaction arguments are prepared to validate whether the system respects the security requirements or not. They include formal and informal proof methods.

The main focus of the framework is to determine security requirements in a more structured way rather than proposing another risk management approach. Trust assumption (Haley *et al*., 2006) and problem frames (Jackson, 2001) are used in this framework to define the security requirements.

In this approach, we see the concepts of *asset, threat* and *harm* which help us to understand the semantics of security risk management artifacts. The mapping of security goals and security requirements of this approach helps to better understand the alignment concepts for modeling languages at the design stage.

2.1.3 The Department of Defense Information Technology Security Certification and Accreditation Process Automation Framework by Lee *et al.*

Certification and Accreditation (C&A) security requirements are expressed in multiple regulatory documents with interdependencies but with different levels of abstraction. This makes these security requirements hard to understand, predict, and control. To address these

issues, Lee *et al.* (Lee *et al.,* 2007) proposed a novel technique by combining software requirement engineering and knowledge engineering[2].

This framework is based on Requirements Domain Model (RDM) which is a hierarchical representation of ontological concepts derived from DITSCAP (Department of Defense Information Technology Security Certification and Accreditation Process) oriented security requirements and policies (Lee *et al.,* 2006). It explains the relationships between security requirements and risk components of the system. It identifies the risk components and map them to concepts in domain specific taxonomies (e.g., threats, assets, vulnerabilities, countermeasures). DITSCAP automation framework follows the steps discussed below to identify security requirements.

1. Security requirements related information about the system and users are gathered by using questionnaires.
2. The answers of the questionnaires are analyzed to expand the set of applicable security requirements including personal controls, logical access controls, network controls and cryptographic controls.
3. After the applicable set of security requirements are defined, their compliance information are gathered. Security requirements and risk relationship model guide the discovery of such compliance information.
4. The relationship between security requirements and the risk relationship model is used to assess the level of compliance of security requirements of the RDM.

This approach (Figure 2.1) shows the relationship between security requirements and risk management. *Risk* related information influences the security requirements. This relationship between security requirement and risk management will help us to understand the risk related concept at the design stage.

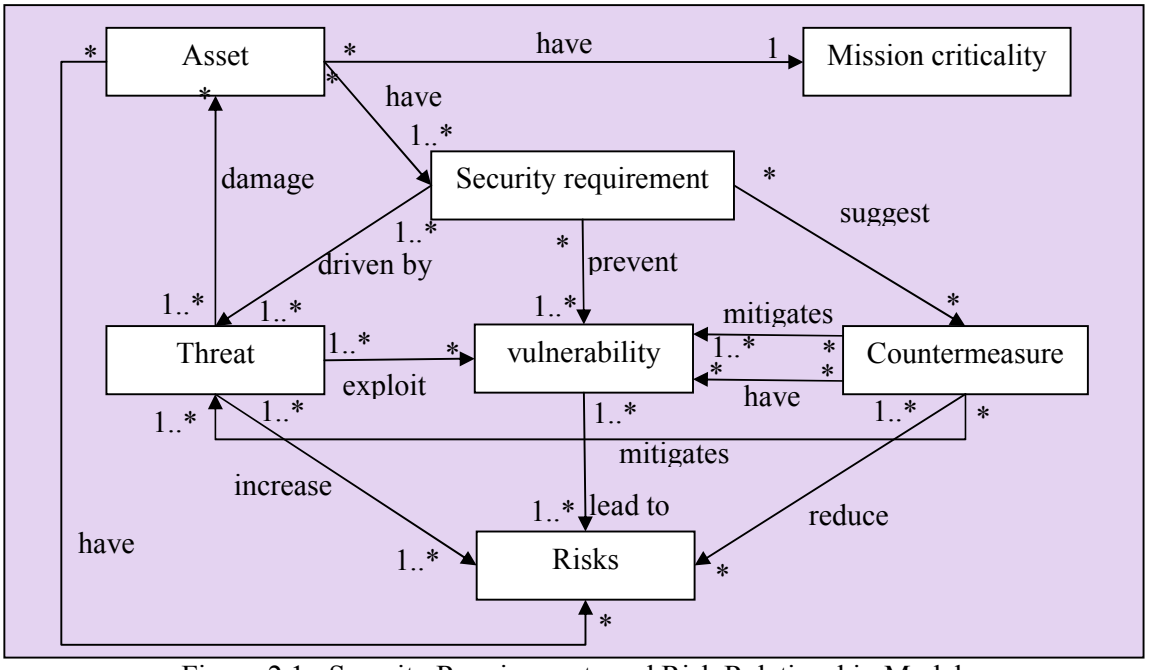

Figure 2.1: Security Requirements and Risk Relationship Model

[2] http://www.epistemics.co.uk/Notes/61-0-0.htm

2.2 Security Risk Management

Security risk management is a process to identify, assess and prioritize risk which is followed by decisions about acceptable risk. The strategies consist of policy options that have varying effects on risk, including avoidance, reduction, reallocation or retention of risk. In the end, an acceptable level of risk is determined and a strategy for achieving that level of risk is adopted. Cost-benefit calculations, assessments of risk tolerance and quantification of preferences are often involved in this decision-making process (Hoo, 2000).

2.2.1 CORAS

CORAS (Braber *et al.,* 2007) is a security risk modeling method for analysis of security threat and risk scenarios. CORAS is mainly focused on the security critical system and it puts more emphasis on information technology security. It uses a customized graphical language (influenced by UML) (Figure 2.2) for communication and documentation of the security risk analysis. It also provides a detail guideline on how to use this language to capture security requirements and model relevant information in different steps of the process. This method also provides a computerized tool[3] to support documenting, maintaining and reporting analysis results through risk modeling.

The CORAS Language

CORAS modeling language supports security risk analysis and several other purposes in each phase of the analysis. A typical security risk analysis is normally structured into five phases: (1) context establishment, (2) risk identification, (3) risk estimation, (4) risk evaluation and (5) treatment identification (Braber *et al.*, 2007). CORAS language consists of five kinds of diagrams to support all these five phases of the risk analysis namely assets overview diagrams, threat diagrams, risk overview diagrams, treatment diagrams and treatment overview diagrams. Their basic building blocks are presented in Figure. 2.2.

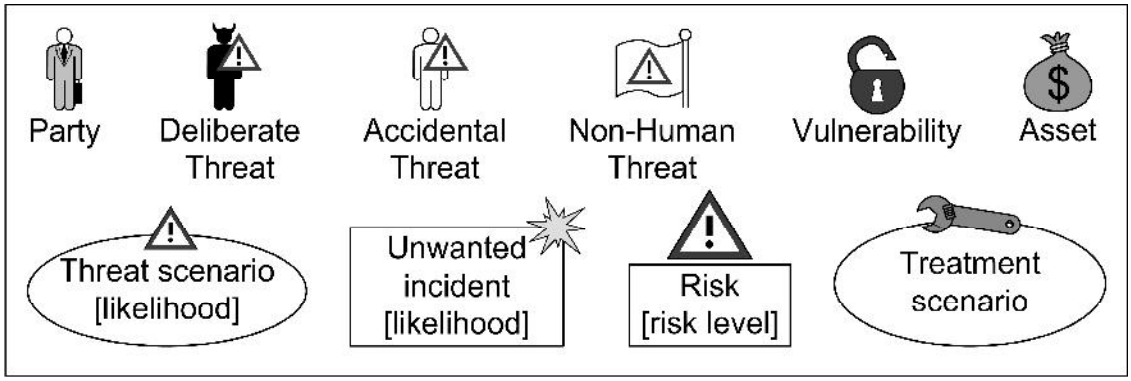

Figure 2.2: Basic Building Blocks of the CORAS Diagrams
(Adapted from Braber *et al.* 2007)

CORAS has developed its own modeling framework with its own concepts. The main concepts proposed by CORAS are,

Assets. Assets are any objects or features of the system which have value to the client. It can be any physical component of the system, services, software or hardware.

[3] http://coras.sourceforge.net/downloads.html

Stakeholders. Stakeholders are the people who are influenced by the system, such as the user of the system, owner of the system and administrator of the system. They are affected by any activity or risk on the system.

Vulnerability. Vulnerability is a weakness of the system or organization.

Threat. Threat may exploit a vulnerability and cause an unwanted incident.

Threat Scenario. It is a sequence of events or activities leading to an unwanted incident.

Unwanted Incident. It is an incident which causes harm to assets and reduces the value.

Risk. A risk is an unwanted incident along with its estimated likelihood and consequence values.

Treatment. Treatments correspond to various choices for reducing risk.

How to Use CORAS
CORAS method consists seven steps to identify, estimate and indulge risk.

Step 1: Introduction. First meeting between analyst and the client. Analyst gets the idea about the goal of the analysis from the client's presentation and discussion.

Step 2: High Level Analysis. Firstly, the analyst presents his understanding from the last meeting's presentation and documents that are made available by the client. Secondly, a very rough security analysis is done. In this analysis the basic threat, vulnerabilities and risks are identified.

Step 3: Approval. In this step, the analyst presents a more refined description of the target that will be analyzed. Here all the assumption and precondition are made about the system to be investigated. Client approves all the documentation at the end of this step.

Step 4: Risk Identification. This step is arranged as a workshop where all the stakeholders and experts are present. The aim of this meeting is to find unwanted scenario, threat and vulnerabilities.

Step 5: Risk Estimation. This is also done as a workshop focusing on estimating consequences of previously identified incidents.

Step 6: Risk Evaluation. The analyst presents the full risk analysis to the client. Client gives his feedback on the analysis and all these feedbacks are incorporated in the analysis.

Step 7: Risk Treatment. In this step the appropriate risk treatment is identified. The cost/benefit issues of the treatments are also discussed in the workshop.

CORAS has its own language and method to deal with security risk. It involves the stakeholders and security experts. There is a continuous communication and feedback mechanism in this method. This approach is also focused on security risk management of information system. The security risk related terminologies and concepts introduced in this method will help us to better understand security risk related concepts at design stage.

2.2.2 Tropos Goal-Risk Model

Tropos Goal-Risk model (Asnar *et al.,* 2007) is built on the core concept of risk and is open to any risk domain such as financial or project management and not limited to information system security. The main objective of this framework is to assess the risk of uncertain events over organization strategies and to evaluate the effectiveness of treatments (Asnar *et al.,* 2008) .

The Tropos Goal-Risk model (GR-Model) is represented by requirements model graph <N,R>, where N stands for nodes and R stands for relations. Any node, N is composed of three constructs, `goal`, `task` and `event`. Goals are strategic interests that actors intend to achieve. Events are uncertain circumstance out of the control of actors that can have an impact on the achievement of goals. Tasks are sequences of actions used to achieve goals or to treat the effects of events.

Key concept of GR-model is to assess risk on the basis of trust (dependency) relation among different actors. To access the risk of the organization GR-Model uses two concepts: delegation and trust. Delegation is used to share the responsibility with other actors. But the delegation sometimes can be vulnerable because if the delegate fails to achieve his goal the whole objective will fail. So, delectation adds *risk* in the system. Trust is used to present the expected behavior from any actor (trustee) by another actor (trustor) in achieving a goal.

The GR-Model consists of three conceptual layers as shown in Figure 2.3 by using our running example (Annex A.1).

- Goal layer analyzes the goals of each actor and identifies which tasks the actor needs to perform to achieve these goals;
- Event layer models uncertain events along their effects to the goal layer;
- Treatment layer identifies and analyses treatments to be adopted in order to mitigate risks.

The final outcome of this model is to find out the level of satisfaction or denial of the goal that is set in the Goal layer.

In short, Goal-Risk model supports modeling by assessing and treating risks on the basis of the likelihood and severity of failures. Risk related concepts discussed in this model will help us to understand the security risk management concepts at the design stage.

2.2.3 Information System Security Risk (ISSRM) Domain Model

ISSRM domain model (Dubois, 2010) is influenced and derived from different security related standards (e.g, ISO/IEC, 2002) and methods (e.g., Braber *et al.,* 2007). To provide a holistic view it accumulates concepts from four main sources, (1) Security Risk Management (RM) Standards, (2) Security related Standard, (3) Security RM Methods, and (4) Security oriented Framework.

ISSRM Domain Model

ISSRM domain model supports definition of security for the key information system constituents and addresses the information system security risk management process at three different conceptual level, *i.e.,* (i) asset-related concepts ; (ii) risk-related concepts; and (iii) risk treatment-related concepts.

Asset-related concepts explain what important assets have to be protected. *Assets* are anything that have values to the organization and the owner or stakeholder wants to protect. The notions that describe the asset-related concepts are *business assets*, *IS assets* and *security criterion*. *Business assets* are plans, processes, information which have value for the organization. *IS assets* are information system components which support business assets. *Security criterion* defines the security needs of the business assets. It is mainly defined as *confidentiality, availability* and *integrity*.

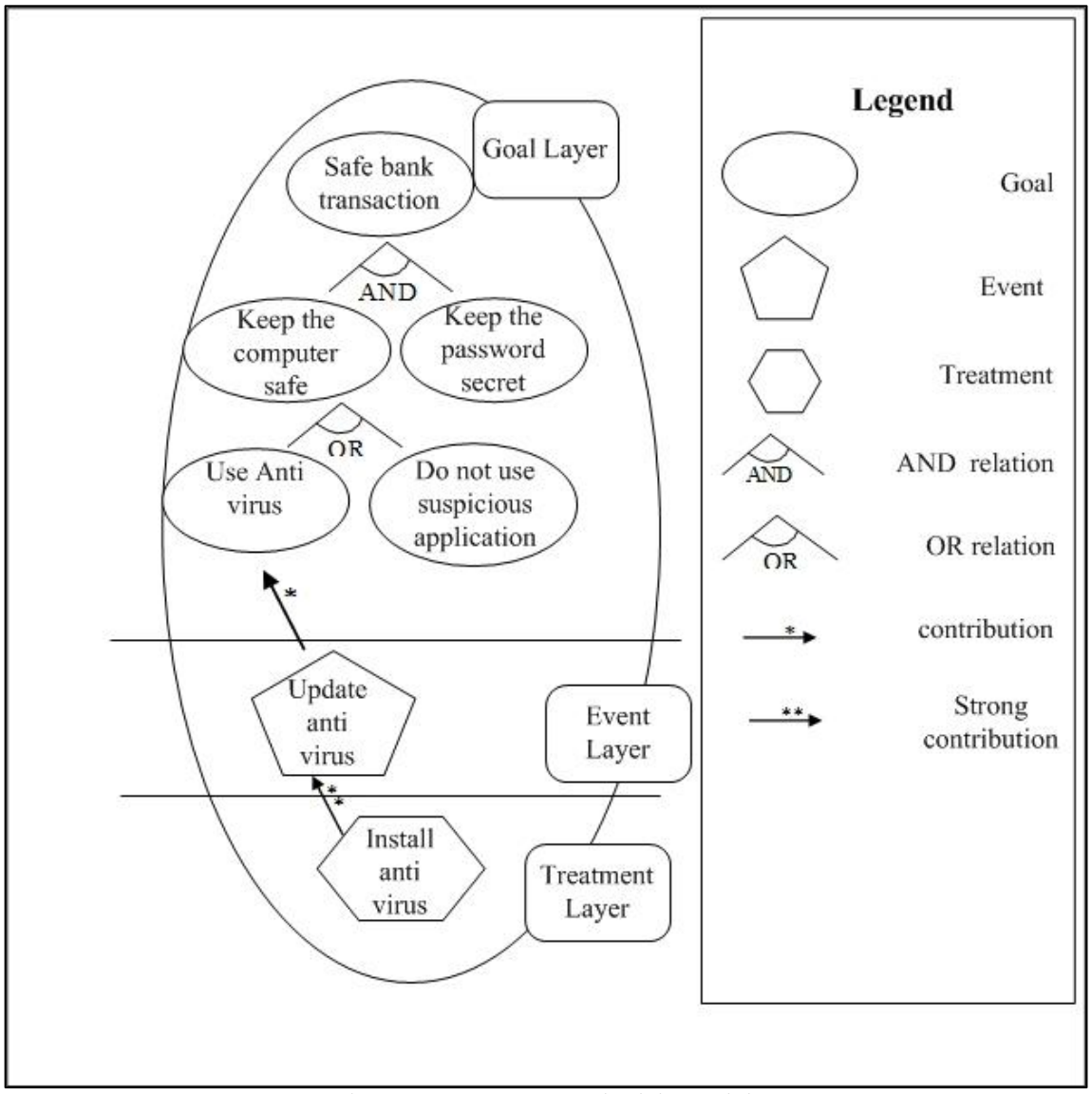

Figure 2.3: Tropos Goal-Risk Model

Risk-related concepts present how the *risk* itself and its components are defined. *Risk* is defined as the combination of a *threat* with one or more *vulnerabilities* which leads to a negative *impact* harming one or more *assets*. *Impact* shows the negative consequence of a risk on *asset* if *threat* is accomplished. *Vulnerability* is expressed as the weakness or any flaws of the *IS asset* or group of *IS assets*. An *event* is composed of a *threat* and one or more *vulnerabilities*. A *threat* is described as a potential attack, which may harm one or more asset

by targeting *IS assets*. An *attack method* characterizes a standard means by which a *threat agent* executes threat.

Risk treatment-related concepts discuss about the *decision*, *security requirement* and *control* to alleviate possible risks. *Risk treatment* is the decision (e.g., *avoidance, reduction, retention* or *transfer*) to handle the identified risk. *Security requirement* is the improvement and fine tuning of the risk treatment decision to mitigate the risk. A *control* provides the means to improve security, defined by implementing the security requirement.

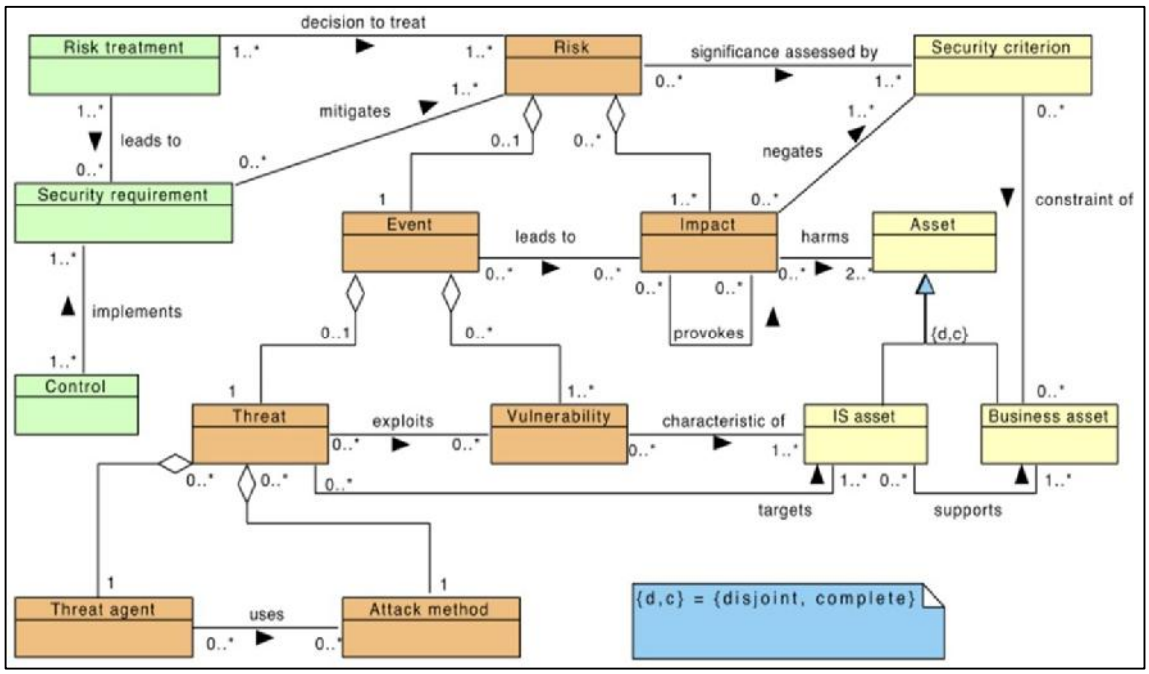

Figure 2.4: The ISSRM Domain Model (Adapted from Mayer, 2009)

ISSRM Process:
ISSRM process is a model based approach which is driven by risk analysis. This is a security requirement engineering process which consists of the following six steps.

Step 1 : Context and Asset Identification. The process begins by defining the organization's context and the identification of its *assets*, both *business assets* and *IS assets*. Description of the organization and its environment are given, focusing on the sensitive activities related to information security.

Step 2: Determination of Security Objectives. In this step, organization defines its security requirements. Security objectives are determined based on the asset identification in the previous step. Security objectives are often defined in terms of *confidentiality, integrity* and *availability* properties of the assets.

Step 3: Risk Analysis and Assessment. In this step, risk identification and estimation are performed. The estimation can be either qualitative or quantitative. The process will go to the next step if analyzed risks have been evaluated against the security needs, which are determined during the second step of the process. Otherwise, the process will go back to Step 1 and has to review the context and asset identification phase.

Step 4: Risk Treatment. Risk treatment can be performed in four different ways, *risk avoidance, risk reduction, risk transfer* or *risk retention*. In *risk avoidance* mechanism, no decision is taken to get involved in any risk situation. *Risk reduction* consists of taking actions to lessen the probability of negative consequences associated with a risk. In *risk transfer* decision, risk is shared with another party. Sometimes it requires some additional security requirements for the third parties. In *risk retention* decision, the burden of loss from the risk is accepted. So, no security measure is necessary in that case.

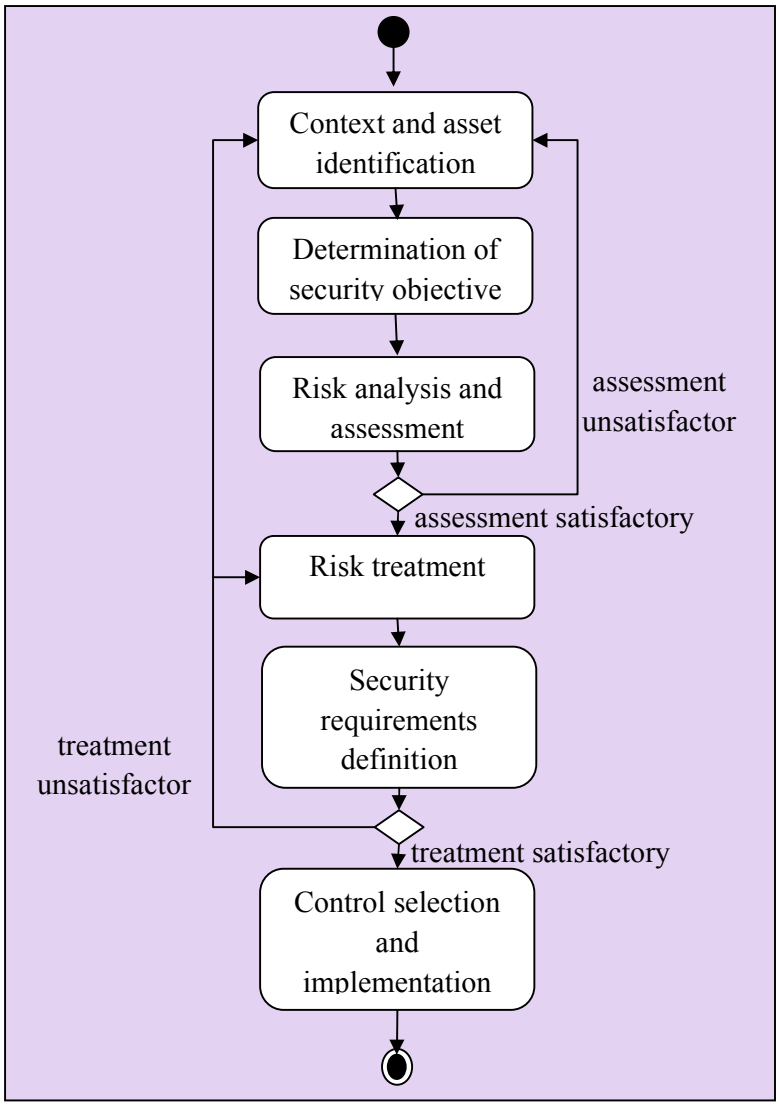

Figure 2.5: ISSRM Process (Adapted from Mayer, 2009)

Step 5: Security Requirements Definition. This step depends on the risk treatment procedure that we have chosen in Step 4. Risk reduction decisions lead to security requirements on the information system to mitigate the risk on the system. Special design decision has to be taken to reduce the risk on the system. Even for risk transfer decision, some special requirement must have to fulfill by the third party. At the end of the security requirements definition step, if they are considered as unsatisfactory, the risk treatment step should be revised, or all of the preceding steps can be revised from the definition of the context and the assets.

Step 6: Control Selection and Implementation. It is a designed means to improve security of the system by taking appropriate security policies, countermeasures and implementation. It is achieved by setting proper security requirements and ensuring the implementation to comply with it.

This process is a repetitive process and the process should be run until the proper security measures are ensured. Lastly, security risks are continuous process and so should be the counter measures.

ISSRM domain model provides its own terminologies and relationship between them. It also defines six step process. It gives detail definition of its terminologies which help us to understand security risk management related concepts. Investigation of this domain model will help us to investigate the security risk management concepts of information system at the design stage.

2.3 Summary

Different security risk management methods (e.g., Braber *et al.,* 2007, Mead *et al.* 2005) and frameworks (e.g., Haley *et al.* 2006, Lee *et al.* 2007) are discussed in this chapter. From this illustration we have got ideas and understanding about concepts and process of different security and security risk management related approaches. From the aforementioned discussions we have come up with different terminologies (e.g., *asset, risk, threat* and *vulnerabilities*) that are used in security risk management domain. In table 2.1, we illustrate these approaches in 5 columns. *Definition of terminologies* shows whether the method or framework is using standard terminologies or using its own terminologies. *Information system based approach* column illustrates whether the approach is focused on information system or not. *Risk based approach* shows whether the approach addresses the risk concept or not. *Supported tools* column shows whether the approach has any tools to support the process. *Used for alignment with modeling language* indicates whether the approach is applied for alignment with any modeling language or not.

From the above discussions and investigation, we have chosen ISSRM domain model to investigate the modeling languages at the design stage. The reasons behind our decision are as follow,

- It has already been used for concept alignment at requirement engineering (Matulevičius *et al.,* 2008)
- It defines the security risk management concepts at three different conceptual levels, which will help us to find and align specific security risk management concepts in any information system.
- It is focused on information system development.

Table 2.1 : A Relative Study of Different Methods and Frameworks

Name of the Method / Framework	Definition of terminologies	Information system based approach	Risk based approach	Supported tools	Used for alignment with modeling language
SQUARE	Definition is set by RE team, experts and stakeholders	Yes	No	No	No
Framework by Haley *et al.*	Standard definition of terminology is used	Yes	No	No	No
Framework by Lee *et al.*	Standard definition of terminology is used	Yes	Have risk related concepts but does not have well defined process	Yes	No
CORAS	Uses its own symbols and terminology	Yes	Yes	Yes	No
Tropos Goal-Risk Framework	Standard definition of terminology is used	It is more general approach.	Yes	No	No
ISSRM domain model	Standard definition of terminology is used	Yes	Yes	No	Yes

CHAPTER 3

Security Risk Management for IS Requirement Engineering

ISSRM domain model has already been used to analyze the security risk management support at the requirement engineering. In this chapter, we will discuss about these (Secure Tropos (Mouratidis, 2005), KOAS(Knowledge Acquisition in automated Specification) with security extension (Lamsweerde, 2004) and Misuse cases (Sindre and Opdahl, 2001)) alignments. We will study these alignment works (e.g., Matulevičius *et al.*, 2008) to understand how they are done (e.g, research method). By understanding these alignment works we can reuse the existing method to analyze security modeling languages at the design stage. This will contribute to answer our first research question: *Is there any domain model which would help to understand security risk management at design stage?*

3.1 Aligning Security Modeling Languages with ISSRM Domain Model

The main objective of the alignment of modeling languages with ISSRM domain model concepts is to verify how these languages support security risk management. Another objective is to help the developers to realize security risk through the modeling language constructs.

For all these three alignments (e.g., alignment of Secure Tropos, KeS and Misuse case), same research method has been used. We will also apply the same research method for the languages at the design stage. The research method consists of three steps (Figure 4.1),

1. Investigate the abstract syntax, semantics and concrete syntax to understand the language.
2. Apply the modeling language according to the ISSRM process (described in Section 2.2.3) by using an example.
3. Analyze the observations (How modeling language construct can be used by following ISSRM process)

These three steps are followed for any particular modeling language by using an example. We will use Meeting scheduler (Feather, 1997), a well-established exemplar in requirement engineering, for this presentation. In Section 4.5, we will summarize these three cases and will provide an alignment table (Table 3.1).

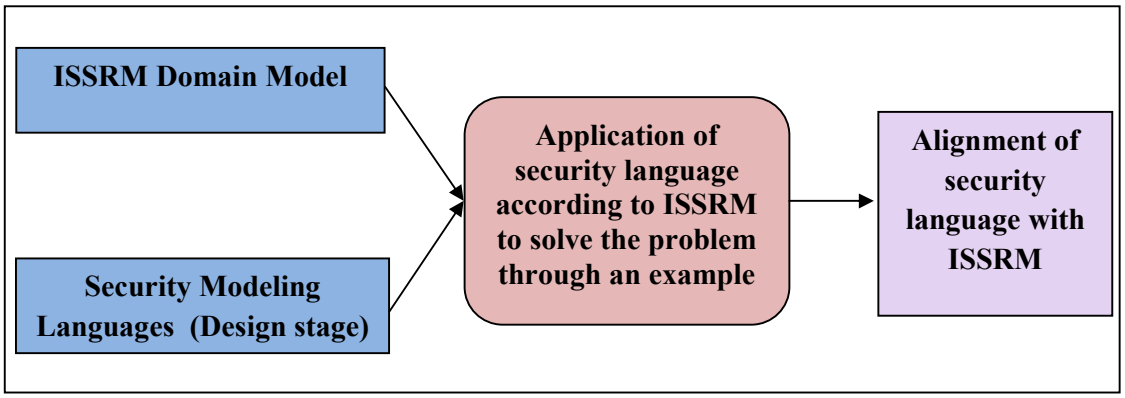

Figure 3.1: Research Method Used for this Work.

3.2 Secure Tropos

Secure Tropos (Mouratidis, 2005) is based on the core concept of Tropos (Bresciani *et al.,* 2004). It is an agent oriented modeling language which is used from the early requirement phase to the architectural and detailed design stage. In this work we will only focus on the early and late requirement stages. Secure Tropos consists of Tropos constructs (`actor`, `goal`, `softgoal` and `resource`) and security constructs (`security constraint` and `threat`). These constructs are connected by relationships: dependency (including sub type of secure dependency) and dependum (e.g., `goal`, `softgoal`, `resource` or `plan`). Semantics of these constructs and relationship can be found in (Mouratidis, 2005).

Mayer has shown how to use Secure Tropos to analyze the security risk and how it aligns with ISSRM domain model in (Mayer, 2009). We will use this illustration in our example (meeting scheduler) to find out the security risk and countermeasures.

We will follow the ISSRM process (described in Section 2.2.3) but will use Secure Tropos constructs to analyze our example (meeting scheduler) here.

1.) Context and Asset Identification. Figure 3.2 shows the content of the meeting scheduler by using Secure Tropos. Here someone (represented as Secure Tropos construct `actor`) will initiate a meeting and the participants (`actor`) will be informed and agreed for the meeting by the `Scheduler` system. The `goal` of the system is to find out an agreement (*Meeting to be scheduled)* for the meeting. This `goal` will be achieved by a `plan` (*obtain agreement*) and finally *agreement* will be established.

2.) Security Objective Determinations. Determination of vulnerable assets and security criteria is supported by `softgoal`. Here, *agreement (business asset)* is confidential and needs to be secured. Thus, we will concentrate on the confidentiality of the agreement.

3.) Risk Analysis and Assessment. Figure 3.3 shows a possible risk event. Here, *Attacker* wants to disclose the agreement. This `goal` of the attacker is to exploit the vulnerability (*Authenticate participant*) of the system. This vulnerability threats `softgoal` (*Only used by participants of the meeting*).

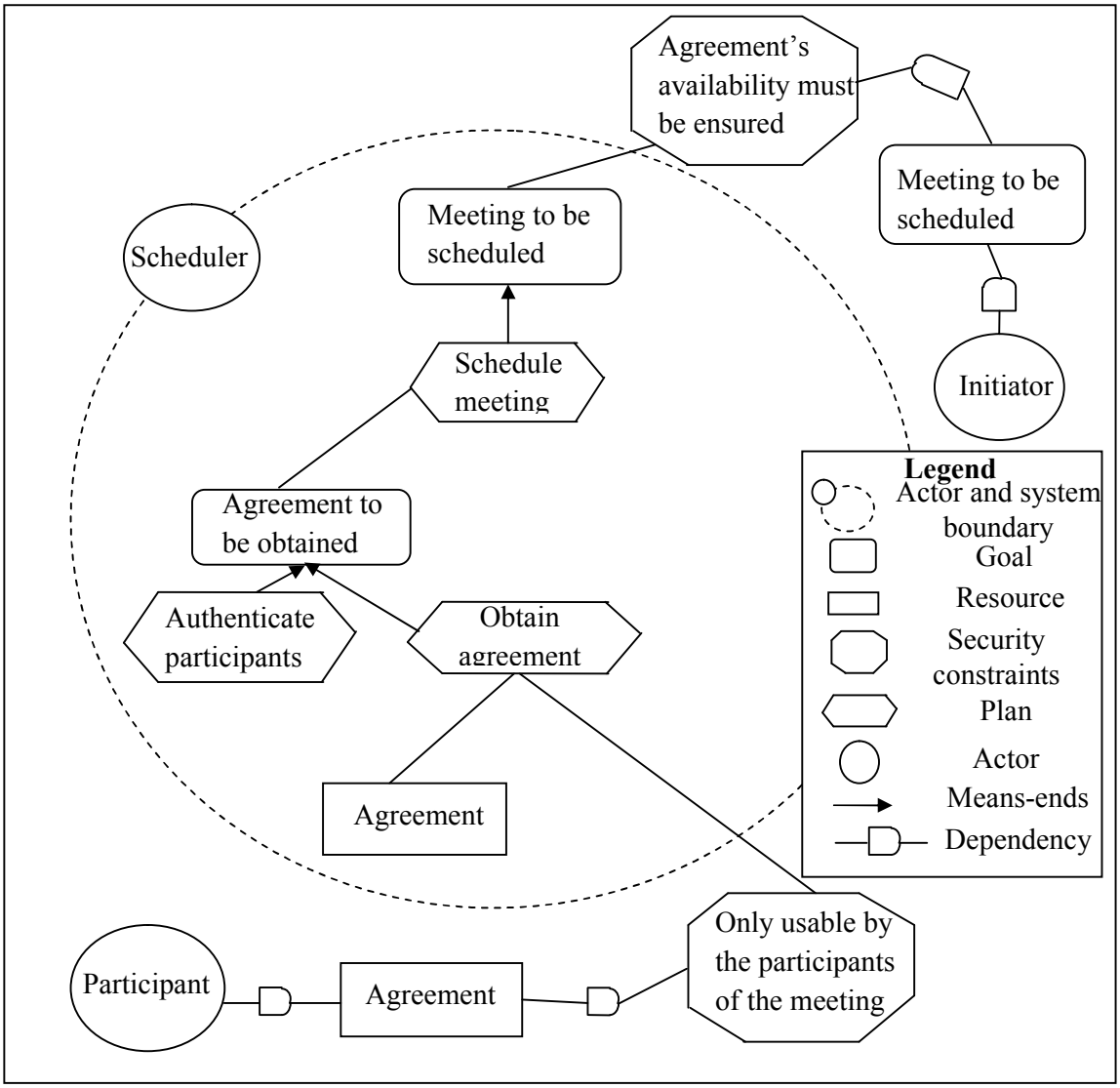

Figure 3.2: Asset Modeling in Secure Tropos

4.) Risk Treatment. Risk treatment is used to take the decision to mitigate the risk. Here we will take a *risk reduction- i.e.,* action to lessen the probability of the negative consequences.

5.) Security Requirements Definition. Based on the risk treatment decision, security requirements are defined by security `goals/plan/resources`. Here, to maintain the confidentiality of the agreement, a `plan` is defined (*Perform cryptographic procedures*) (Figure 3.4). Even if the attacker gets the agreement information he cannot read the agreement because it is encrypted. Thus, this `plan` ensures the security requirement (confidentiality) for the agreement.

6.) Control Selection and Implementation. Secure Tropos does not suggest any techniques to select and implement controls.

Alignment of ISSRM and Secure Tropos. The detail alignment between these two approaches is presented in (Mayer, 2009). They are aligned in the following way and also presented in Table 3.1.

ISSRM asset-related concepts*: business assets* and *IS assets* are represented by `actor`, `goal`, `resource` and `plan constructs` of Secure Tropos. *Security criterion* is expressed by `security constraint` and `softgoal` constructs of Secure Tropos. About risk related concepts, risk cannot be represented by any constructs of Secure Tropos, but Secure Tropos can represent *event* (as a `threat` construct) and *impact* (as directed contribution between `threat` and `softgoal`) by its constructs. ISSRM *threat agent* is defined by Secure Tropos `actor` and *attack method* is defined by `plan` (with attack relationship). ISSRM *threat* is represented by Secure Tropos `goals/plans`. *Vulnerability* does not match with any constructs but it is possible to show the *IS assets* that have `vulnerability points`. ISSRM risk treatment related concepts are represented by Secure Tropos `actors`, `goals`, `resources`, `plans` and `security constraints`.

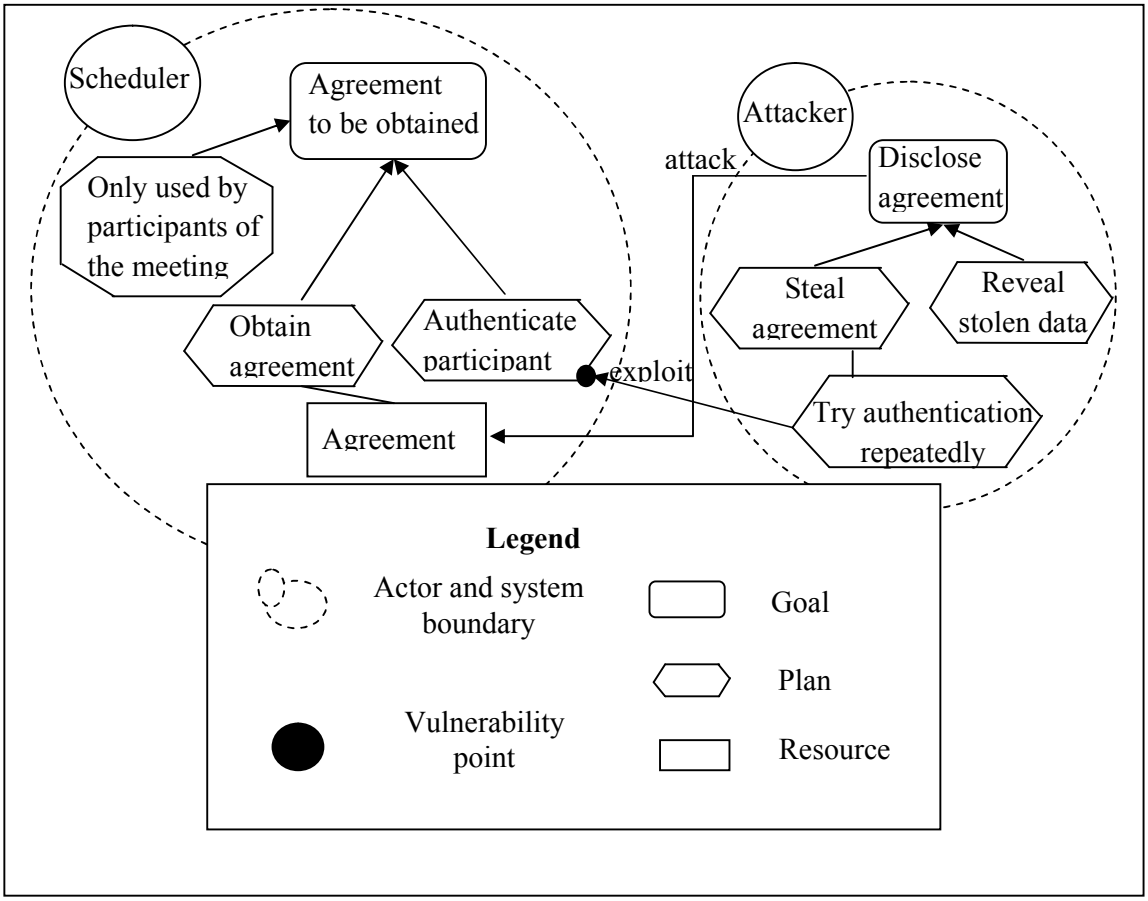

Figure 3.3: Threat Definition Using Secure Tropos

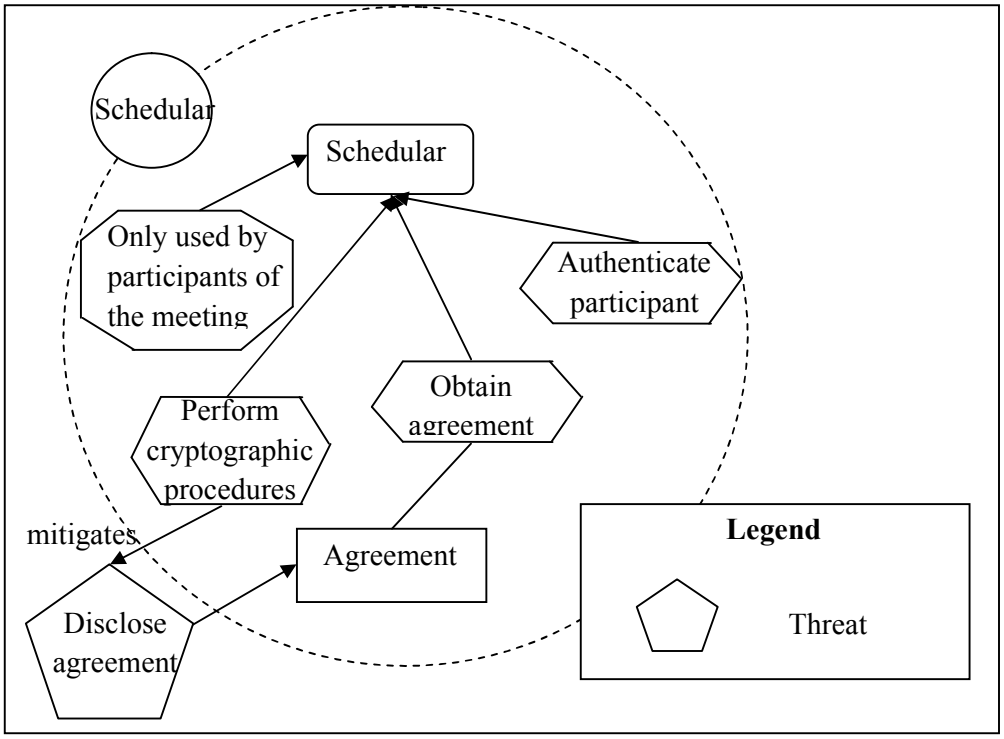

Figure 3.4: Risk Treatment Using Secure Tropos

3.3 KAOS extension to Security (KeS)

The KAOS (Knowledge Acquisition in automated Specification) approach consists of a modeling language, a method and a software environment. The main purpose of KAOS is to ensure that high-level goals are identified and progressively refined into precise operational statements. Along this process, various alternative goals and responsibility assignments are considered until the most satisfactory solution is chosen. KAOS extended to Security (KeS) has been introduced in (Lamsweerde, 2004). The main objective of KeS is defined by `goal`. The `goal` can be decomposed into several sub goals. These sub goals are again refined till the requirement is defined. The detail procedure to meet the goal is defined by an operational model. Risks to the system are represented by anti goal model. And the countermeasure of the anti goal model is defined by a new security requirement. The action of the model is defined by using two constructs `achieve` and `avoid`. `Achieve` means the system must have to achieve this goal and `avoid` means system must have to resist it from this goal.

1.) Context and Asset Identification. Figure 3.5 shows the content of the meeting scheduler by using KeS. The main goal studied here is *Achieve [arrange meeting]* which is characterize by `domain property` (*Only in business hours).* The goal is also refined as `subgoal`(*Accept the invitation, Meeting invitation and Avoid [InvitationReceived ByCrook]).* The detail procedure of the information system is given as operational model in Figure 3.5. The `goal` (*Meeting invitation)* is performed by *Initiator.* He also performs other operations (*Select the time, Send invitation* and *Select participants).* The `objects` are used to support `goals`, here `object` is *Database of participants.*

2.) Security Objective Determinations. The determination of security objectives is done in the same model (Figure 3.5) and generally in the same time as the elicitation of other goals. Security criteria are defined by using `Avoid`. Here, we have seen *Avoid*

[InvitationReceived ByCrook] as security objective which ensures the integrity of the system. This security objective can be achieved by two alternative `goals` *Avoid [AccessToSystem ByCrook]* and *Avoid [LoginInformationKnown ByCrook]*.

3.) Risk Analysis and Assessment. Figure 3.6 shows a possible risk event. Here, *Attacker* wants to know the login information of the legitimate user of the system. Risk analysis is done by creating an anti goal model. Like goal, anti-goal is also defined and decomposed as sub goal (e.g., *Achieve [UsernameKnownByCrook], Achieve [PasswordKnownByCrook]*) until reaching anti requirement (e.g., Achieve [UseSocialEngineeringtoLearnPassword]) assigned to anti-agent (Attacker) in Figure 3.6. Vulnerabilities are also identified in anti-model, like *EmployeesNotSecurityAware*. The operation model is also defined to meet the security requirements.

4.) Risk Treatment. In KeS, risk treatment is defined through the countermeasure chosen for handling the anti-model and its associated vulnerabilities and anti goals. In our example, *risk* (here vulnerability) *avoidance* is chosen as countermeasure.

5.) Security Requirements Definition. New security goals are emerged from this countermeasure. A new goal model is thus built, with additional security goal(s), requirement(s) and/or expectation(s). A new requirement called *PerformAwarenessTraining* is added to the goal model presented in Figure 3.7. This requirement is assigned to the *Security officer* (`agent`).

6.) Control Selection and Implementation. KeS does not suggest any techniques to select and implement controls.

Alignment of ISSRM and KeS. The detail of the concept alignment between KeS and ISSRM domain model can be found in (Mayer, 2009). ISSRM asset related concepts are represented by KAOS `goal`, `requirement` and `expectation` (both *business asset* and *IS asset*). In fact, KeS does not differentiate *business asset* and *IS asset*. `Operation` and `object` are also used to present *asset*. Security criteria are expressed by `goal` and `object attributes`. *Threat agent* is presented by `anti-agent` and *action method* by operationalisation, domain and required conditions and operation. *Vulnerability* is defined by the `domain property`. At higher level of abstraction `anti goal` represents *event* and at lower level (realization) of abstraction it (in combination with anti-requirements and anti-expectation) represents *threat*. *Security requirement* is represented by `security goal`. This goal can be further refined by security requirement and expectation.

3.4 Misuse Cases

Concept alignment between Misuse cases/Misuse case diagram (will discuss in Section 4.3) and ISSRM domain model is presented in (Matulevičius *et al*, 2008). The assessment is done by analyzing the Misuse cases meta-model and textual explanations.

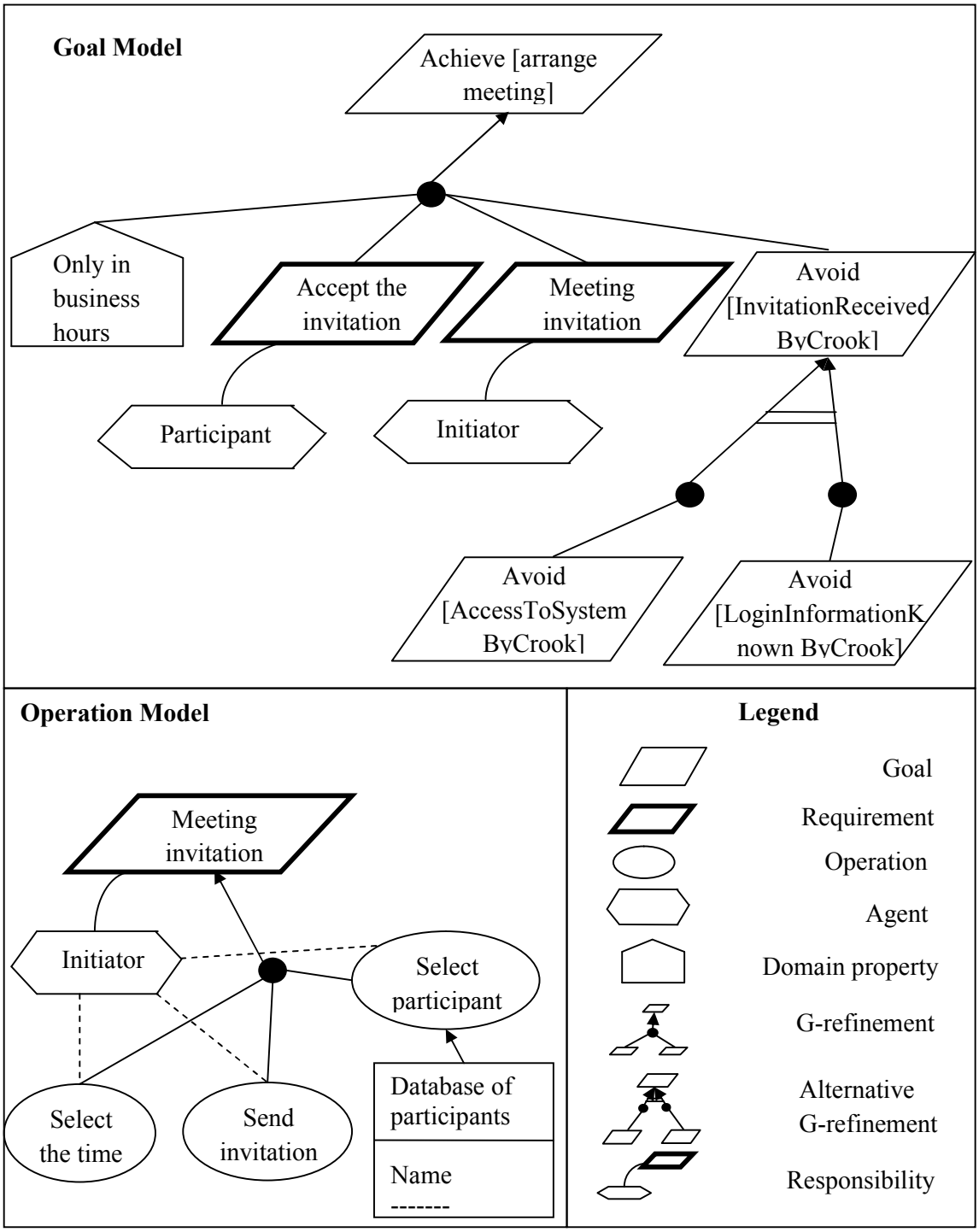

Figure 3.5: Asset Modeling Using KeS

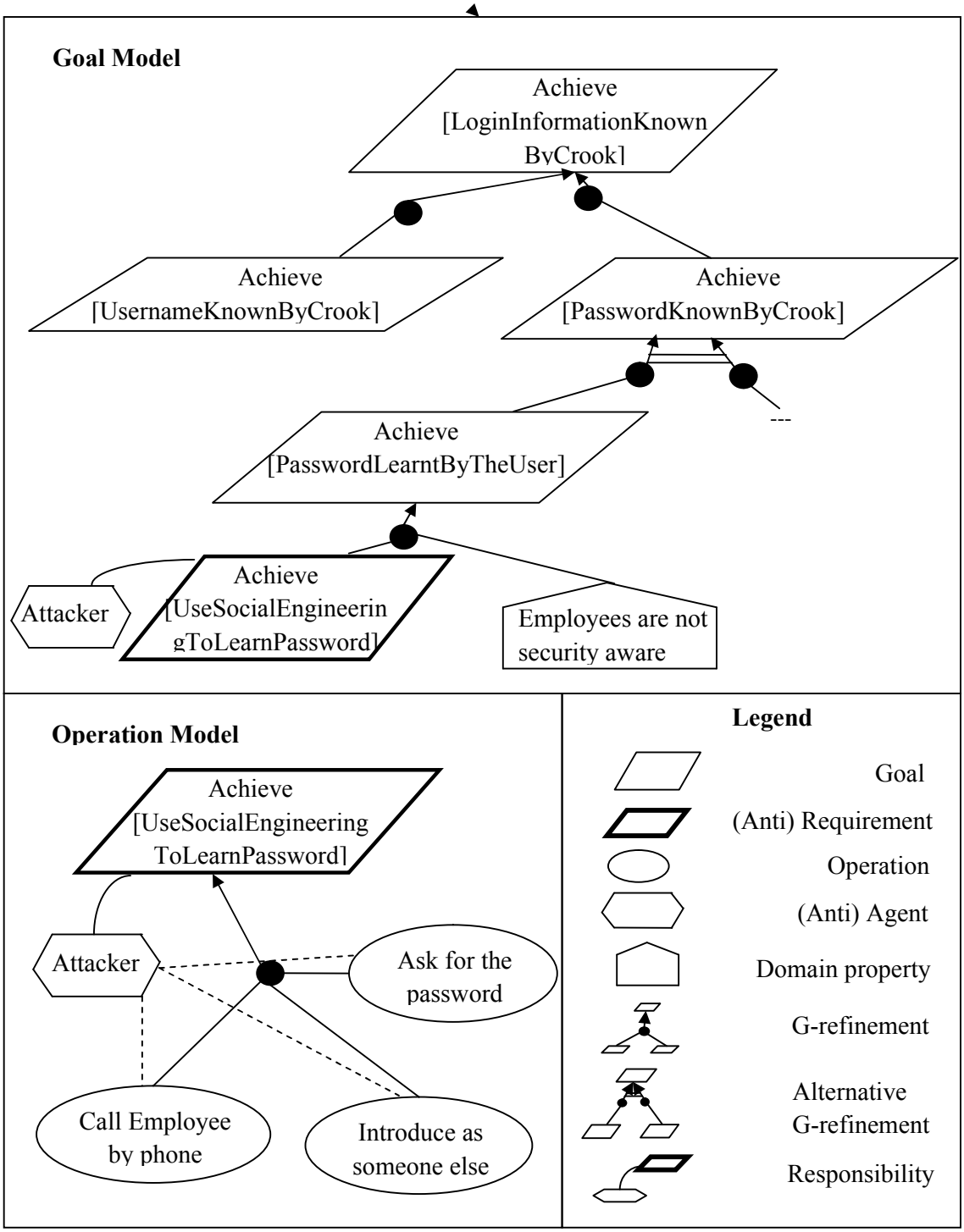

Figure 3.6: Threat Definition Using KeS

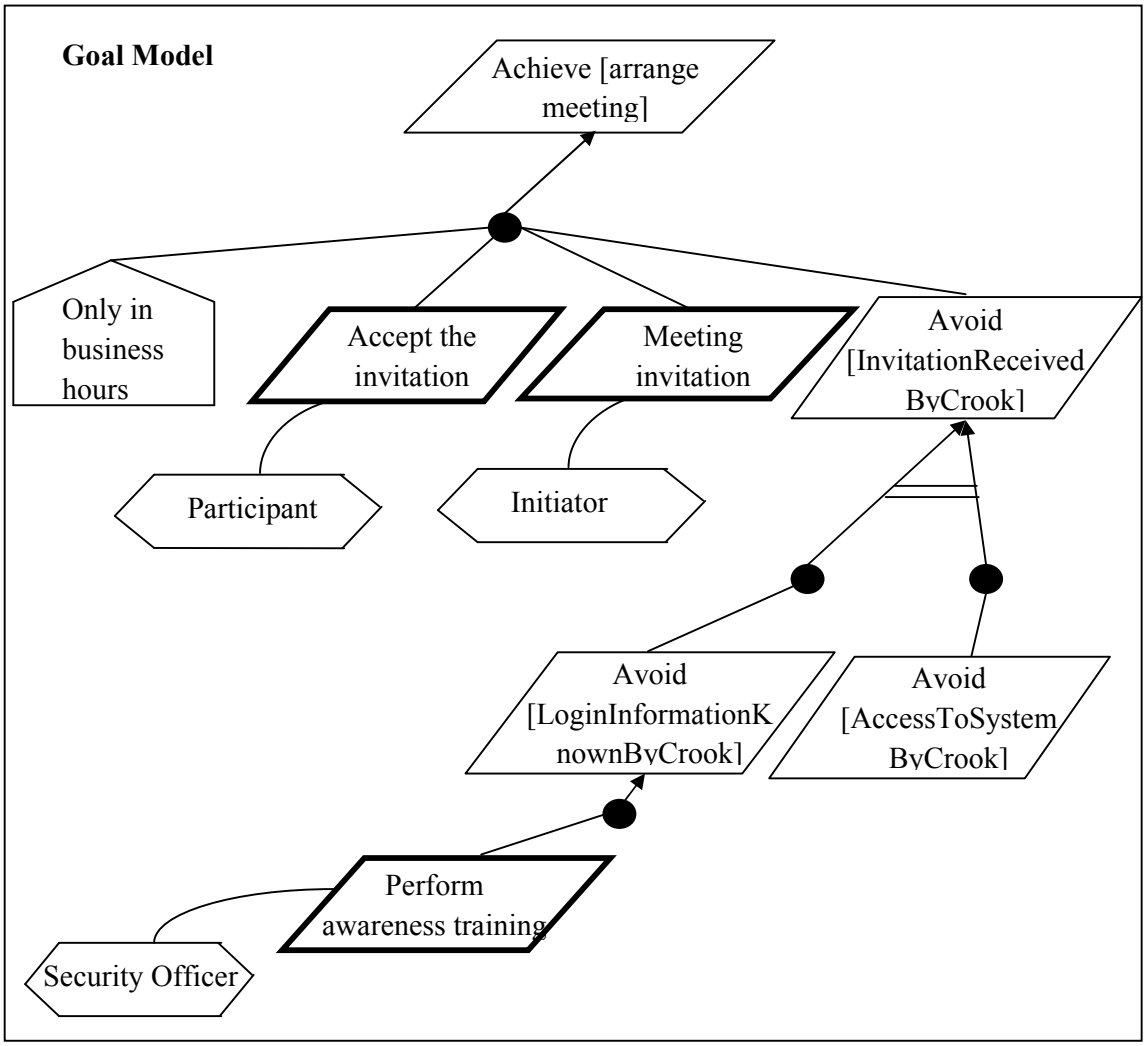

Figure 3.7: Risk Treatment Using KeS

1.) Context and Asset Identification. In Figure 3.8, we have modeled the meeting scheduler example. Here, the element of the meeting scheduler is represented by `use case` in the diagrams. For example, *obtain available dates* in the Misuse case diagrams is the *asset* for the meeting scheduler.

2.) Security Objective Determinations: Determination of vulnerable assets and security criteria is not supported by use cases. Use cases can only be used to reason about security criteria without showing them in a diagram. In our example, we concentrate on three security objectives: (i) availability of agreement; (ii) confidentiality of agreement, and (iii) integrity of agreement.

3.) Risk Analysis and Assessment. Misuse case diagrams involves a `misuser` (*attacker*) (Figure 3.9). The Attacker targets the availability with the `misuse case` (*Make agreement unavailable)*. It `threatens` the `use case` *(Store agreement)*. The Attacker targets the confidentiality and integrity by other `misuse cases` (*Change the date of agreement* and *Disclose agreement*). *Disclose agreement* `includes` two other `misuse cases` (*Steal date* and *Reveal stolen date*) **.**

4.) Risk Treatment. The Misuse case diagrams do not suggest any risk treatment. `Security use cases` can be selected as the *risk reduction* decision out of four possible *risk treatment* processes based on general security risk management process.

5.) Security Requirements Definition. Misuse case diagrams has `mitigates` syntax which provides security requirements for the system. In the running example, *Check participant identity* is a security use case which mitigates the risk (*Disclose agreement)* (Figure 3.10).

6.) Control Selection and Implementation. Misuse case does not suggest any technique to select and implement controls.

Alignment of ISSRM and Misuse Cases. ISSRM asset related concepts are expressed by standard `use case` construct. *Business asset* and *IS asset* are represented by `actor, use case` and `system` construct of Misuse case diagrams. Risk related concepts: *threat agent* is presented by `misuser` and `misuse case` represent *attack method*. So, *threat* is defined by the combination of `misuser` and `misuse case.` Risk treatment related concepts are supported by security use case in the use case diagrams.

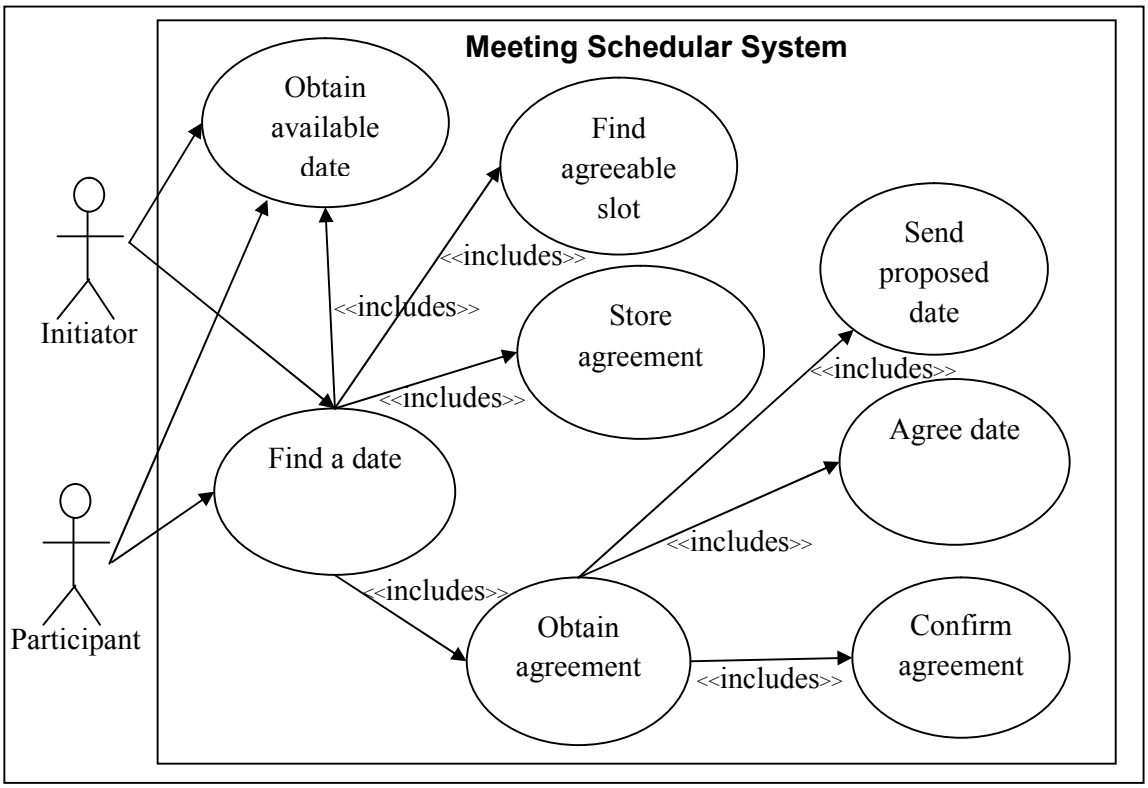

Figure 3.8: Asset Modeling Using Misuse Case Diagrams

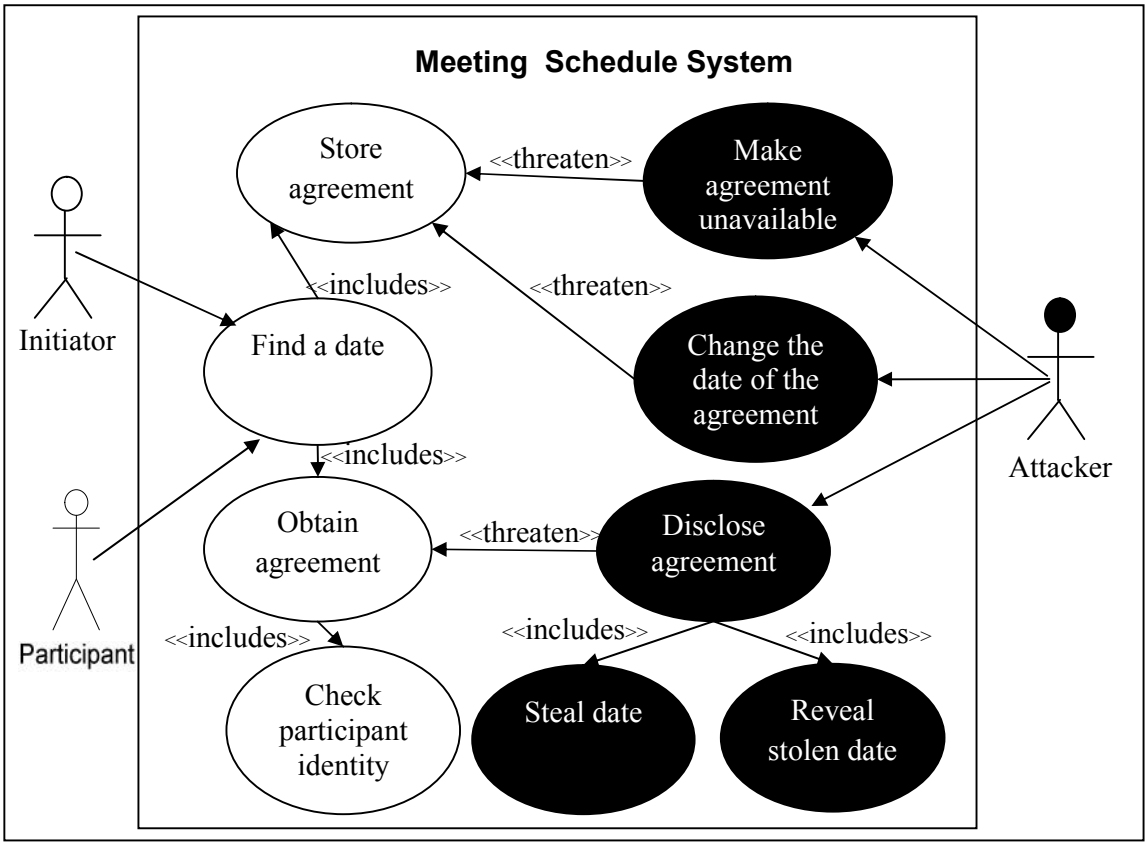

Figure 3.9: Threat Definition Using Misuse Case Diagrams

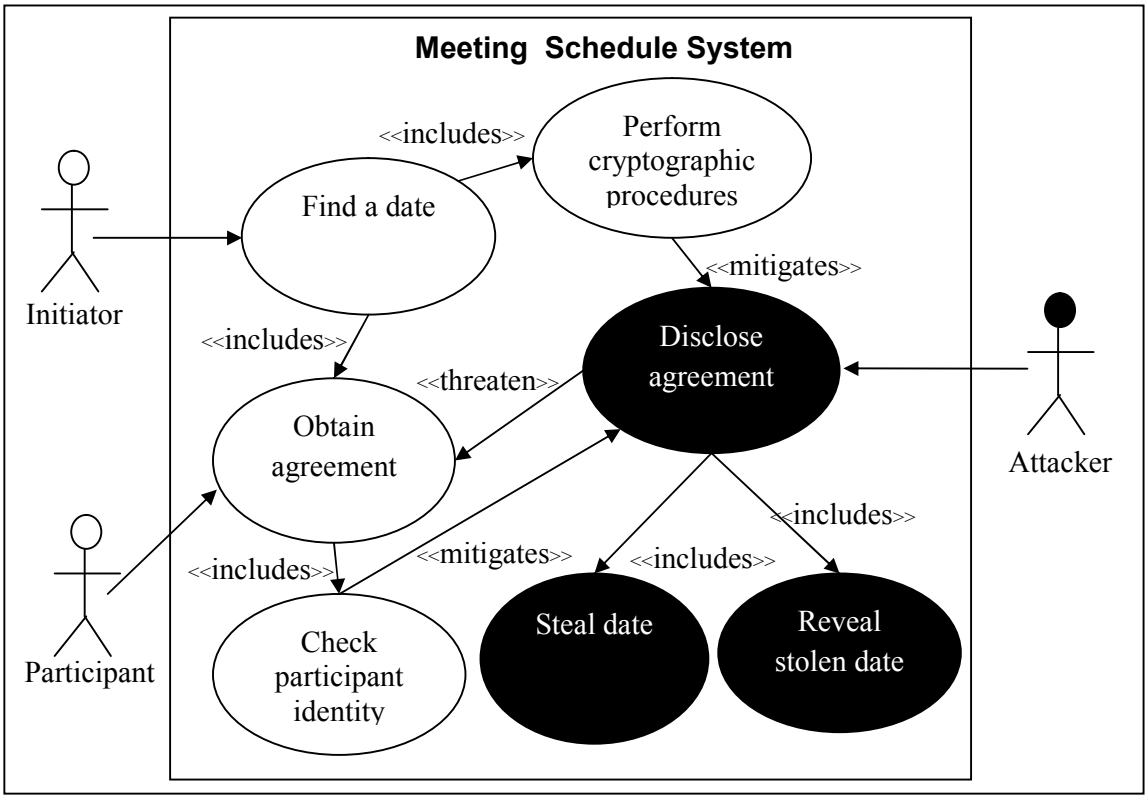

Figure 3.10: Security Requirements Using Misuse Case Diagrams

3.5 Summary

In this chapter, we have discussed about the research method that is used for the alignment of the requirement engineering language and ISSRM domain model. The discussion in this chapter proves that this method can identify requirement engineering constructs which can be mapped with ISSRM domain model concepts. So, we will follow the same method for the alignment at the design stage.

Table 3.1 shows the summery of the concept alignment. Here, we present how each ISSRM domain model concept is represented by these three modeling language constructs. Empty place in the alignment table shows that there is no appropriate construct of the modeling language which can represent that ISSRM concept. In our alignment work in Chapter 5 and Chapter 6, we will also follow this table structure to show our results.

Table 3.1: Concept Alignment of Security Languages with ISSRM Domain Model

ISSRM Concepts		Secure Tropos Constructs	KeS Constructs	Misuse Cases Constructs
Asset related concepts	Asset	Actor, Goal, Softgoal, Plan, Ressource	Goal, Requirement, Expectation, Operation, object	Actor and Use case
	Business asset			
	IS asset			
	Security criterion	Security constraint, Softgoal	Goal, Object attributes	
Risk related concepts	Risk			
	Impact	Contribution between threat and softgoal		
	Event	Threat	Goal, Requirement, Expectation (in anti-model)	
	Threat	Goal, Plan		Misuser and Misuse case
	Vulnerability		Domain property	
	Threat agent	Actor	Agent	Misuser
	Attack method	Plan, relationship attack	Operationalisation + Domain and required conditions + Operations	Misuse case
Risk treatment related concepts	Risk treatment			
	Security requirement	Actor, Goal, Softgoal, Plan, Ressource, Security constraint	Goal, Requirement, Expectation	Misuse case
	Control	New model implementing security components	New model implementing security components	New model implementing security components

CHAPTER 4

Security Modeling Language for System Design

Modeling languages (e.g., UML (Object Management Group (OMG), 2004), Tropos (Bresciani, 2004)) help to design, analyze and validate enterprise and application level architectural models. To integrate security features throughout the software development cycle, modeling languages have introduced security extension (e.g., Secure Tropos (Mouratidis, 2005), SecureUML (Lodderstedt *et al.,* 2002), UMLSec (Jurjens, 2002) etc). We will investigate concepts and ideas of these extensions and will try to answer our second research question: *What are potential security modeling languages at the design stage to support security risk management?*

UML has become the *de facto* industry standard for specifying software intensive system development. In this chapter, we will only look into security extension of UML based diagrams. We will investigate syntax, semantics and the process of the security extension of these modeling languages by using a running example (Appendix A.1).

4.1. Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a visual modeling language which is used for specifying, documenting and constructing the artifacts of an object-oriented software intensive system. UML provides a visual means to present requirements, design and deployment of software system.

UML offers three major advantages: visualization, complexity management and communication. UML is used to support software development methodology(e.g., Waterfall methodology[4], Agile methodology[5]) in different ways (e.g., by using Usec ase, Activity diagrams) but it does not provide any development methodology. UML uses the following notations and semantics to describe software development process:

- **Use Case Model** - describes the user interaction with the system. It also defines the system boundary. It is mainly used in requirement phase.
- **Class Model** - usually used in design phase and describes the classes and objects which build the system.
- **Communication Model** - describes how objects in the system will interact with each other to get work done.

[4] http://www.buzzle.com/editorials/1-5-2005-63768.asp
[5] http://www-01.ibm.com/software/rational/agile/

- **Activity Diagrams** – This model is used to present dynamic behavior of the system. It describes the workflow of the system.
- **The Physical Component Model** - illustrates the hardware and software components that make up the system.

UML diagrams are mainly divided into two different views of a system model:

- **Static (or structural) view:** emphasizes the static structure of the system using objects, attributes, operations and relationships. The structural view includes class diagrams and composite structure diagrams.
- **Dynamic (or behavioral) view:** emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams.

We will look into different security extensions of UML here. Abuse cases and Misuse cases are security extension of Use case and used at requirement engineering. Mal activity diagrams are proposed by extending Activity diagram and used in late requirement and at the design stage. Both UMLSec and SecureUML are used at the design stage and are presented by using *UML profile.*

4.2 Abuse Case Diagrams

McDermott has proposed Abuse case diagrams/Abuse cases (McDermott and Fox, 1999) to bridge the gap between security domain and software requirement engineering. Abuse case diagrams are defined as interactions between one or more actors which result into harmful impact on the system or to other actors of the system. Another aspect of Abuse case diagrams is that it should describe the privilege that is used to complete the abuse case. Any abuse can be happened by demolishing the total control of the system or by just taking control of some part of it. Abuse case can also be accomplished by compromising any legitimate actor's credentials. Because of the range of privileges, the Abuse case diagrams should be described from minimum privilege violation to maximum level of violation.

Main Concepts of Abuse Cases

- Abuse case diagrams are a family of transactions between one or more actors and a system, that results in harm.
- UML-based use case diagrams.
- Potentially one family member for each kind of privilege abuse and for each component that might be exploited.
- Includes a textual description of the range of security privileges that may be abused.
- Includes a description of the harm that results from an abuse case.

Example: We will apply Abuse case diagrams process (McDermott and Fox, 1999) on online banking system example (described in Appendix A.1). In Figure 4.1, we see that actors (*Hacker* and *Malicious client*) are interacting with the online banking system. The interactions are *steal login information, steal personal information* and *change banking policy* (`abuse case`). These interactions are presented as `abuse case` in Abuse case diagrams and result into harm to the system (Online Bank).

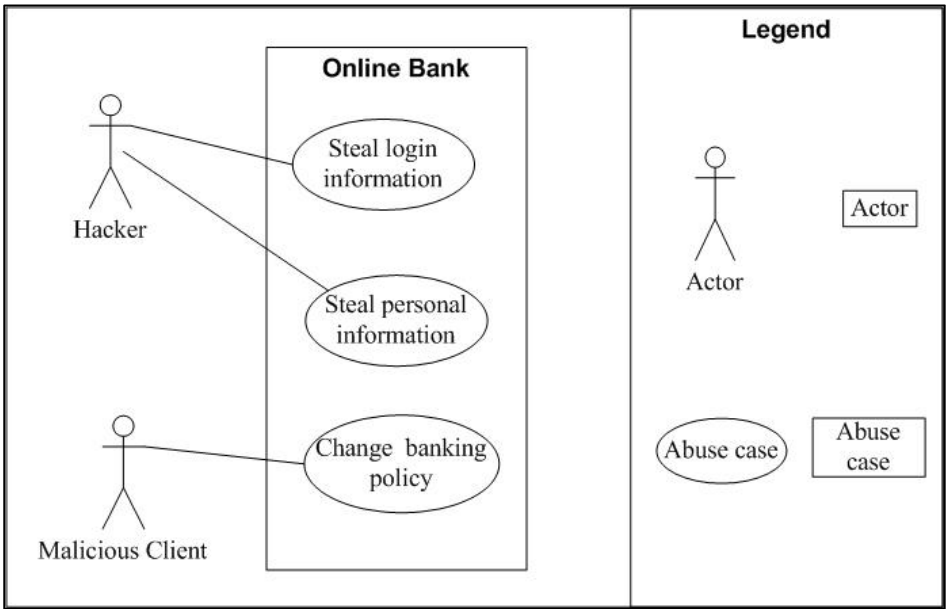

Figure 4.1: Abuse Case Diagrams

Textual description is used to give details about Abuse case diagrams. The textual description is composed of two components: actor description and abuse case description. Three main characteristics are defined related to actors: `resources` (e.g., software, hardware, other tools), `skill` (e.g., technical skills related to information security) and `objectives` (e.g., financial benefit, vandalism, terrorism etc). An example of textual description of `abuse case` (*steal login information*) is proposed for *Hacker* in Figure 4.2.

Actor Description	Abuse Cases Description
Hacker Resources: The hacker has software, hardware and necessary tools, and internet connection and enough financial support and time to break the security of the system. Skill: The hacker has enough technical skill about information security, network protocols and cryptography. Objective: The hacker's interest is to gain financial benefit by transferring money to his own account or to another person's account who has assigned him.	**Steal login information** Harm: The hacker can transfer money to his own account by stealing login information. Privilege range: By using the clients privilege, the hacker can see the transaction, transfer money, change the login information, and pay bills etc. Abuse interaction: by stealing the login information the hacker can steal many confidential information about the user. If the client use the same password for other services then those application/system will be at risk

Figure 4.2: Textual Descriptions of Abuse Case Diagrams

4.3 Misuse Case Diagrams

Misuse case diagrams/ Misuse cases(Sindre and Opdahl, 2001) are proposed by extending traditional Use case diagrams by introducing negative use cases. In another way, it can be seen as use case from a hostile actor's perspective. It is also represented by two ways: textual specification and graphical diagrams (Figure 4.3). Use case diagrams only gives an overview of the system functionality, a relatively detailed description of the scenario is provided by an associated textual description. This textual description is based on a template (presented in Sindre and Opdahl, 2001) to be filled out by an analyst. The template is extended to make it suitable for describing Misuse cases by supporting detailed elicitation and analysis of security threats. Misuse cases reuse the concepts existing in Use case diagrams namely `actor`, `use case` and the associated relationships: `threaten`, `include` and `mitigate` relationship.

Like Abuse case, Misuse case is also used at the requirement engineering. To see how the Misuse case concepts support the ISSRM domain model concepts, a concept alignment is presented in section 3.3. Main difference between an Abuse case diagrams and Misuse case diagrams is that Misuse case diagrams keeps the normal use case in the diagrams.

Example: Figure 4.3 presents a Misuse case diagrams based on online banking system (Annex A.1). Misuse case diagrams comes with a security requirement process, which outcomes into the elicitation of suited security requirements. The process consists of five steps (Sindre and Opdahl, 2005). We will use this process in the example to see how Misuse cases work.

1.) Identify Critical Assets. *Login to the system, Do the transaction* and *Set banking rules* are the `use case` of the Misuse case diagrams in Figure 4.3. These use case can also be seen as the critical assets of this system.

2.) Define Security Goal. Security goal is to maintain the confidentiality and integrity of the login information, transaction information and banking rules that needs to be protected from `malicious actor` (e.g., *Hacker*).

3.) Identify Threats. *Steal login information* which is a `misuse case`, is treated as the threat to the online banking system. It is shown by relationship associating `threaten` to `use case` (*Login to the system).*

4.) Identify and Analyze Risks. *Steal login information* is achieved by using another `misuse case` (*Use fake website).* This is represented by using a relationship association `include`.

5.) Define Security Requirements Graphical Misuse. Security requirement is set to achieve the security goal. To do this a new `actor` (*Security Official*) is introduced. He will *Establish security training plan* which `includes` a *`use case (security use case )`Create awareness among users*. This will `mitigate` the security risk of the online banking system.

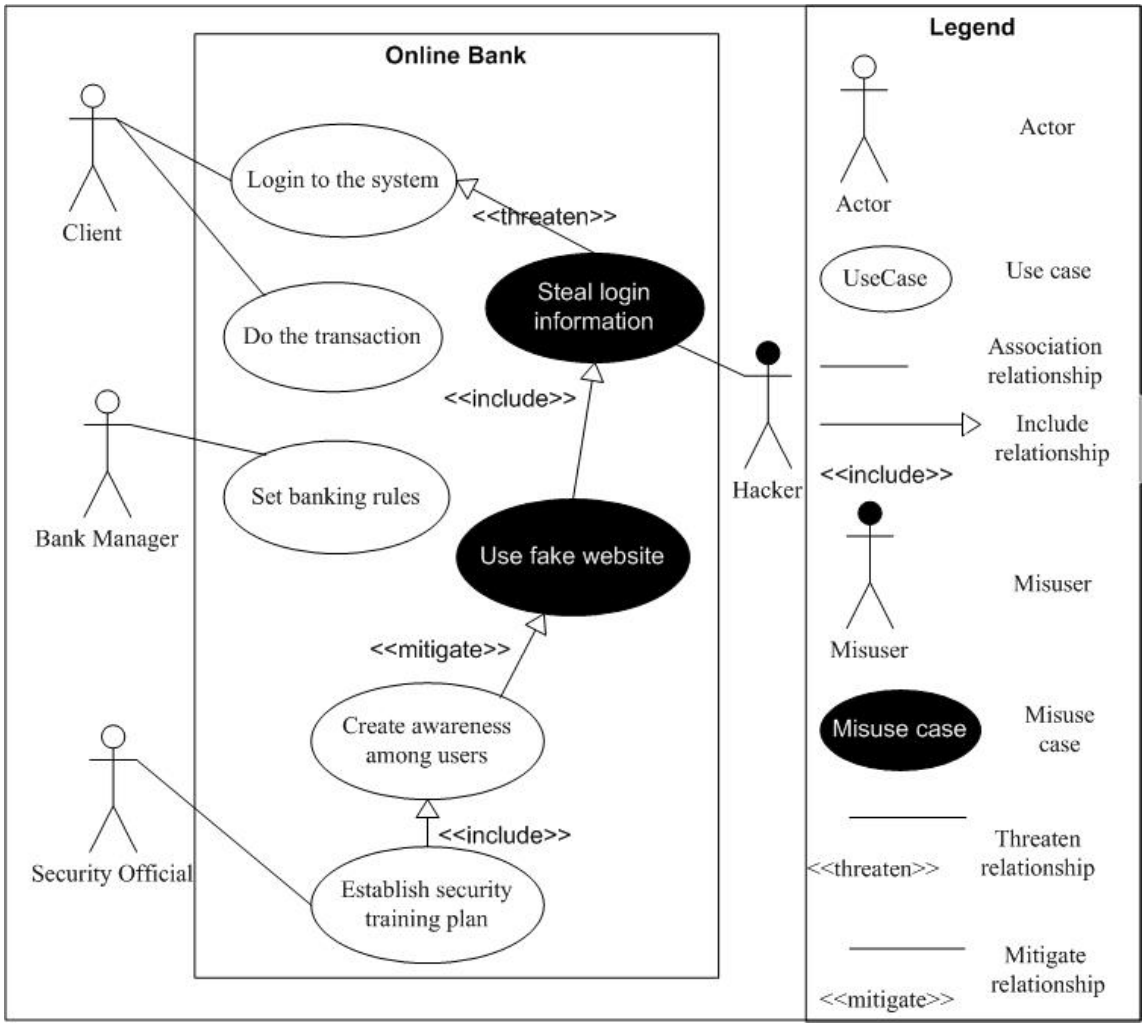

Figure 4.3: Misuse Case Diagrams

4.4 Mal Activity Diagrams

Mal activity diagrams (Sindre, 2007) is derived by extending the concepts of Activity diagram. It deals with the behavioral aspects of the security problems. Basic way to build a Mal activity diagrams is to build a normal process first then add the mal activity (unwanted behavior) against this process. This concept is similar to the Misuse case diagram in the way that it also allows to put the defensive process (e.g., mitigation activity) to the diagrams. It adds some extra concepts other than the ordinary activity diagrams, such as `Mal-Activity`, `Mal-swimlane` and `Mal-decision`. These concepts are just opposite of the normal activity diagrams constructs. It also defines `MitigationActivity` and `MitigationLink` to show the mitigation process.

Unlike Misuse cases and Abuse cases, Mal activity diagrams do not have any defined process about how to use Mal activity diagrams. Sindre has used these concepts on 46 cases presented in (Mitnick, 2002). The concept alignment between Mal activity diagrams and ISSRM domain model is presented in Chapter 5.

Example: Figure 4.4 shows how Mal activity diagrams can model online banking system example(Annex A.1). It shows the attack process by a *Hacker* (`Mal-Swimlane`) to the online banking system with the help of a *Fake website*. The first `Mal-Activity` of the attacker is to *Send email with a fake website link.* The *fake website* is a look alike of the target website and maintained by the *hacker*. The client will *Receive the malicious mail* and based on his decision to open or to delete the e-mail, the fake website will be opened in his browser or the process will stop there. If the client puts the login information in the fake website, the hacker will *Collect the login information*. Depending on the nature of the password (e.g., one time password), the `Mal-decision` indicates the ability of the *Hacker* to collect useful login information. A way of representing mitigation option is also proposed in the Mal activity diagrams. On the right hand side of Figure 4.4, the mitigations (`MitigationActivity`) are shown in a separate column and inserted into the attack process with dashed arrows (`MitigationLink`). The *Security module* is there to mitigate the security risk. The proposed mitigation activities are *Enable mail filtering* before *Receive the malicious e-mail* and *Enable anti-malware* before *Collecting the login information*.

Mal Activity Diagrams Meta Model

We propose a meta model for Mal activity diagrams in Figure 4.5 (based on the available literature (Sindre, 2007) on Mal activity diagrams). Mal activity diagrams start with an `InitialState` (starting point) and finishes with a `FinalState` (end point). Mal activity diagrams consist of three kinds of activities: `Activity`, `Mal-Activity` and `MitigationActivity`. `AnySwimlane` holds all the constructs of the Mal activity diagrams. `AnySwimlane` includes `Swimlane` and `Mal-swimlane`. `Swimlane` contains `SwimlaneElement`, which is composed of `Activity`, `MitigationActivity` and `Decision`. An `activity` is the specification of a parameterized sequence of behavior. A `MitigationActivity` shows the improvement of the process to avoid `MalciousActivity`. `Decision` illustrates branching based on order rejected or order accepted conditions.

`Mal-swimlane` includes `Mal-swimlaneElement`, which consists of `Mal-activity` and `Mal-decision`. `Mal-activity` is the inverse of normal activity. These activities are done by malicious actor to harm the normal business process. `Mal-decision` is a decision which is made with a malicious purpose.

About the cardinality: One `AnySwimlane` can include many `AnyState`. One `Swimlane` can include many `SwimlaneElement` and one `Mal-swimlane` can include many `Mal-swimlaneElement`. The elements are complete and disjoint.

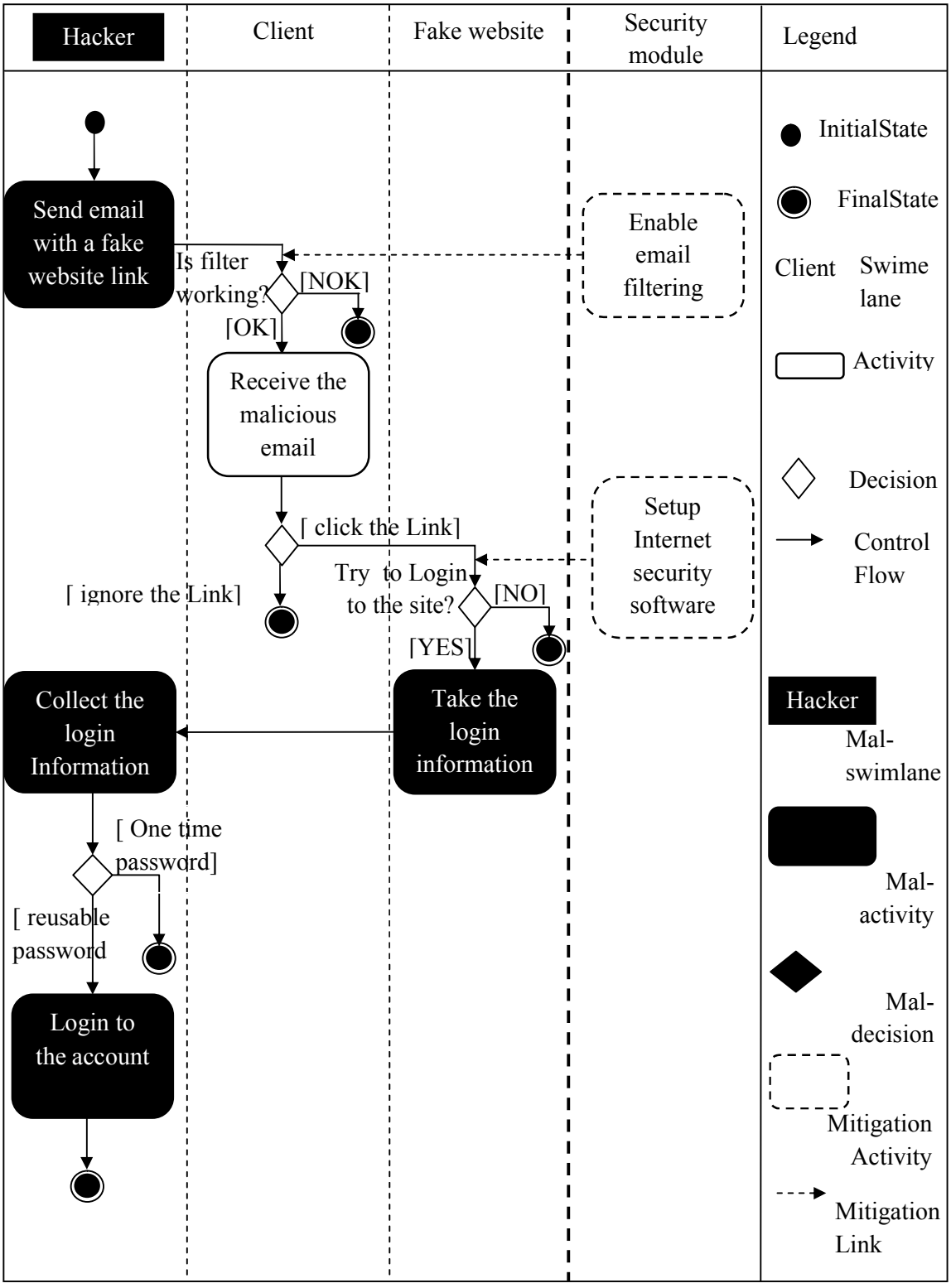

Figure 4.4: Mal Activity Diagrams

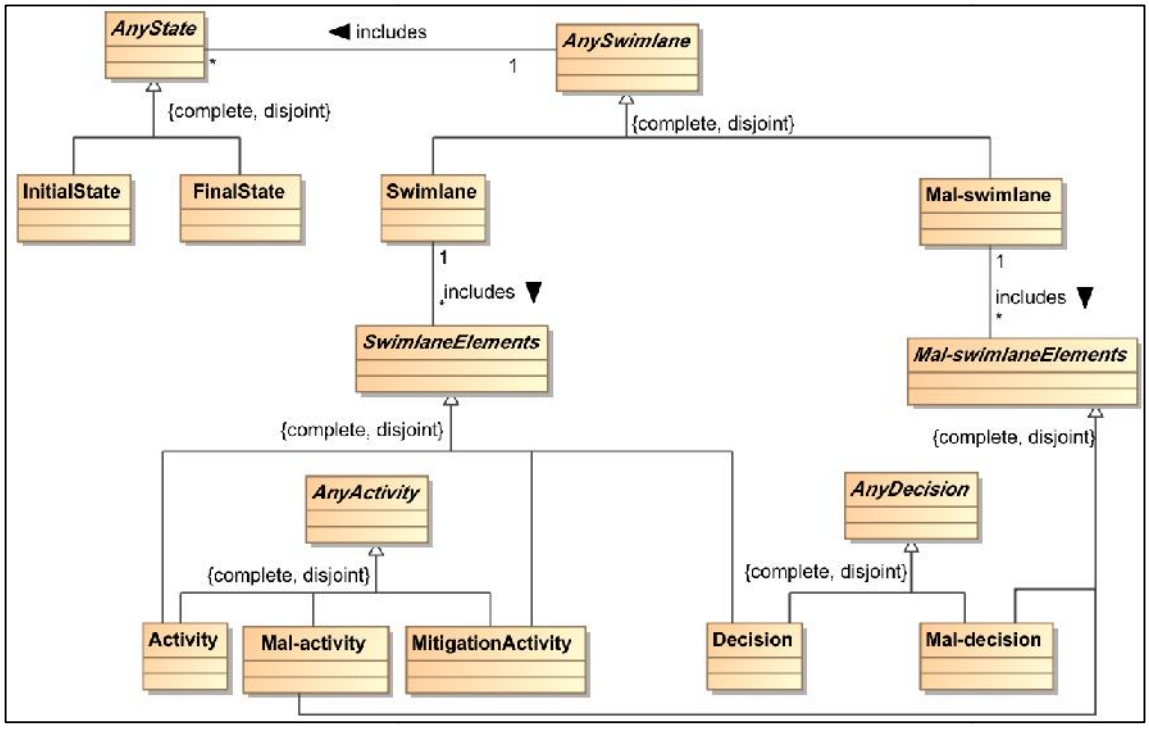

Figure 4.5: Meta Model of Mal Activity Diagrams

Figure 4.6 shows the control flow of the Mal activity diagrams. The process stars with `InitialState` then it goes to `AnyActivity` which consists of `Activity`, `Mal-activity` and `MitigationActivity`. Different `Activity`, `Mal-activity` and `MitigationAction` are connected by `ControlFlow`. `MitigationLink` suggests where in the process the `MitigationActivity` would be added. `AnyDecision` is also used in the flow with the `Activity` and `Mal-Activity` when there is a need for any choice between two alternatives.

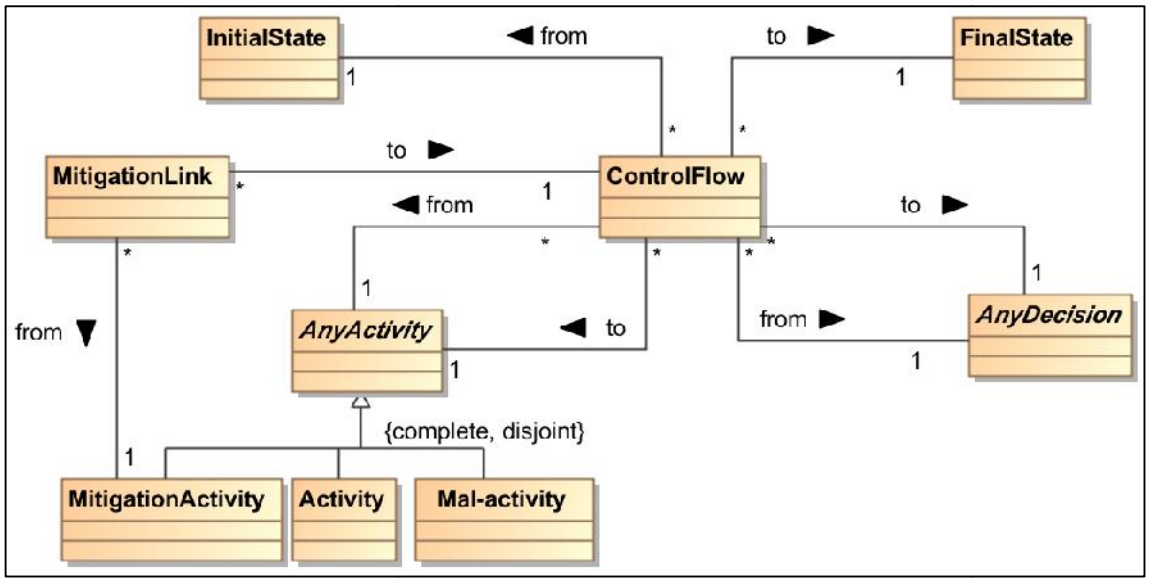

Figure 4.6: Control Flow of Mal Activity Diagrams

4.5 UMLSec

UMLSec (Jurjens, 2002) is a *UML profile* based extension of the UML diagrams for integrating security information. The profiles are defined by a set of `stereotypes`, `base class`, `tag` and `constraints`. `Stereotypes` define new types of modeling elements by extending the semantics of existing types or classes in the UML meta model. Their notation consists of the name of the stereotype written in double angle brackets << >> and attached to the extended model element. This model element is then interpreted according to the meaning described to the stereotype. One way of explicitly defining a property of the model element is by attaching a `tagged value` to it. Another way of adding information to a model element is by attaching `constraints` to refine its semantics. UMLSec defines 21 stereotypes together with their tags and constraints to support security requirements (Jürjens, 2004). Main strength of UMLSec is that it is very precisely defined and has formal semantics. It evaluates UML specification for vulnerabilities and encapsulate security engineering pattern.

Like Mal activity diagrams, UMLSec is also used at the design stage of the information system development process. We will see how its concepts help to design a secure system with the help of our running example (Annex A.1).

Example: We will use UMLSec notations to specify and design secure online banking system. As discussed in the example (Annex A.1), online banking is a web based application. The main banking application is hosted in a secure server and clients get access to the banking service by using web browser. In Figure 4.7 we see that web server gets the password from the client application, which is running on client's browser. As this data is very confidential so it should be passed through a secure link. A class with a stereotype `<<secure link>>` and `adversary tag` is used to define the appropriate level of security. But as the data is going through ordinary Internet communication link. It needs extra (e.g., confidentiality, integrity) protection.

In open Internet the secrecy of the data may be compromised in an adversarial environment. This provokes extra security requirement which should be realized at the design stage. To fulfill these requirements encrypted and data security stereotype are defined. A class with a stereotype `<<encrypted>>` ensures that all the information between the browser and the online banking server will be encrypted. So, adversary could not interpret or alter the message. But it does not ensure integrity. To ensure integrity a class with a stereotype `<<data security>>` is used. `<<data security>>` also ensures authenticity by using `integrity` and `authenticity` tags.

4.6 SecureUML

SecureUML (Lodderstedt *et al.,* 2002) is based on role-based access control with additional support by specifying authorization constraints. It embeds role-based access control (RBAC) policies in UML class diagrams using a UML *profile*. It defines a vocabulary for annotating UML based models with information relevant to access control. SecureUML defines a vocabulary for expressing different aspects of access control, like `role`, `permission` and `user-role assignment`.

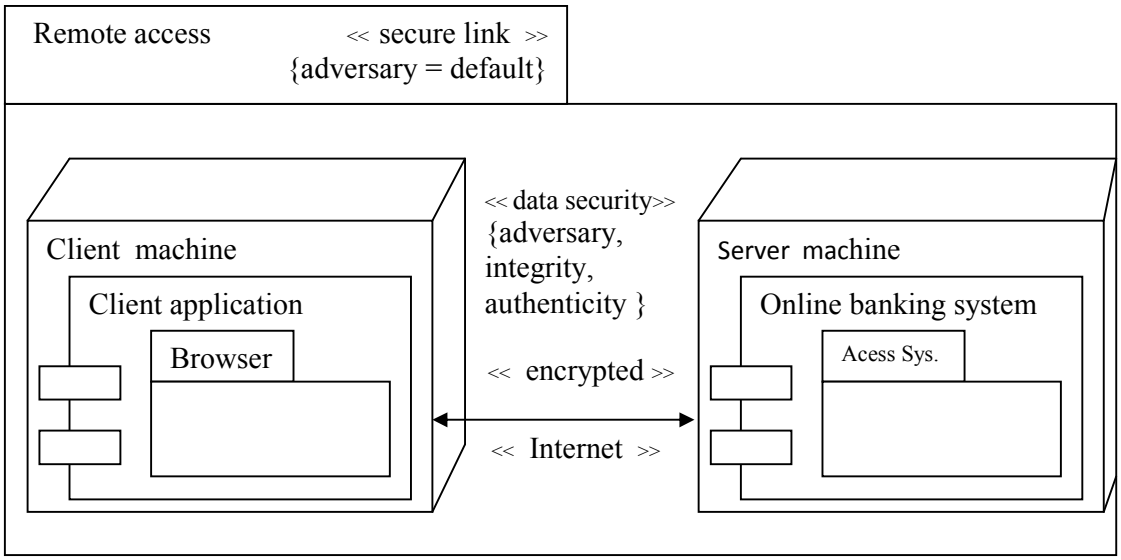

Figure 4.7 : UMLSec Diagrams

`Authorization constraint` defines the precondition for granting access to any operation. Such constraints are expressed using Object Constraint Language (OCL)[6]. `Authorization constraint` gives SecureUML the flexibility to define and verify the access decision on dynamically changing data. Like UMLSec, SecureUML is also used at the design stage. We will thoroughly investigate different concepts of SecureUML and will also try to find the concept alignment with ISSRM domain model in Chapter 6.

Example: We will use SecureUML model on online banking system (Annex A.1). SecureUML uses its main concepts (e.g., `user`, `role`, `ModelElement` and `permission`) to design secure software application. Figure 4.8 shows a scenario of money transfer. SecureUML is used by the following steps,

1.) Identify Users. In this system *Alice* is the client of this online bank. So, *Alice* can be seen as the user (a class with a stereotype <<`user`>>) of this system.

2.) Identify Application Roles. Online banking system defines a role (a class with a stereotype <<`role`>>) *account holder* for the client to get the service from the system.

3.) Map Users into Roles. *Alice* (`User`) is mapped with the *account holder* (`role`)in the system.

4.) Identify Resources. *Bank account* (a class with a stereotype <<`ModelElement`>>) is treated as the resources for this example.

5.) Identify Actions. Stereotype permission (an association class with stereotype <<`permission`>>) is used to define the action for the system. It defines different actions by using attributes `ActionType`. In this example, *viewAccount:read* and *transfermoney: update*can be seen as action.

[6] http://www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-04-05/OCL-Essay.pdf

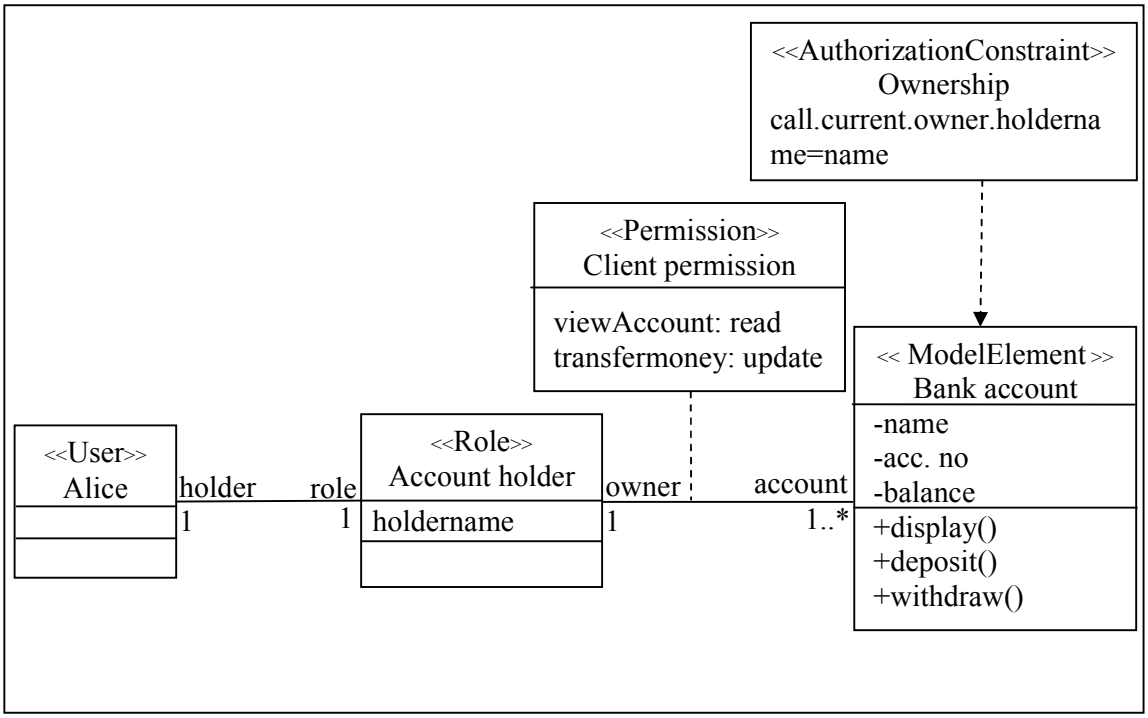

Figure 4.8: SecureUML Diagrams

6.) Identify Authorization Constraints: `authorization constraints` must be satisfied before any authorization to apply. It can be seen as the precondition. In this example, transaction can be done only by the account holder. It is ensured by the constraint *Ownership*.

7.) Account for Cardinality: SecureUML also shows cardinality. Here, it shows one-to-many relationship between *Account holder* and *Bank account* **.**

SecureUML Metamodel

The SecureUML meta model (Figure 4.9) is represented by the role-based access control (RBAC) concepts. It introduces three new concepts `User, Role` and `Permission`. Due to the design decision, every UML model element is represented as a protected resource and defined as a class with a stereotype `<<ModelElement>>`. `<<ResourceSet>>` is used to represent a user defined set of model elements. An association class with a stereotype `<<Permission>>` is a relation object connecting a `role` to a `ModelElement` or a `ResourceSet`. The semantics of a permission is defined by the `ActionType (`attributes of `Permission)` elements used to classify the permission. Every Action-Type represents a class of security relevant operations on a particular type of protected resource. Action types give the developer a vocabulary to express permissions at a level close to the domain vocabulary. A class with a stereotype <<Role>> is a job or function within an organization. It combines all privileges needed to fulfill the respective job or function.

An `AuthorizationConstraint` is a part of the access control policy of an application. It expresses a precondition imposed on every call to an operation of a particular resource, which usually depends on the dynamic state of the resource, the current call, or the environment.

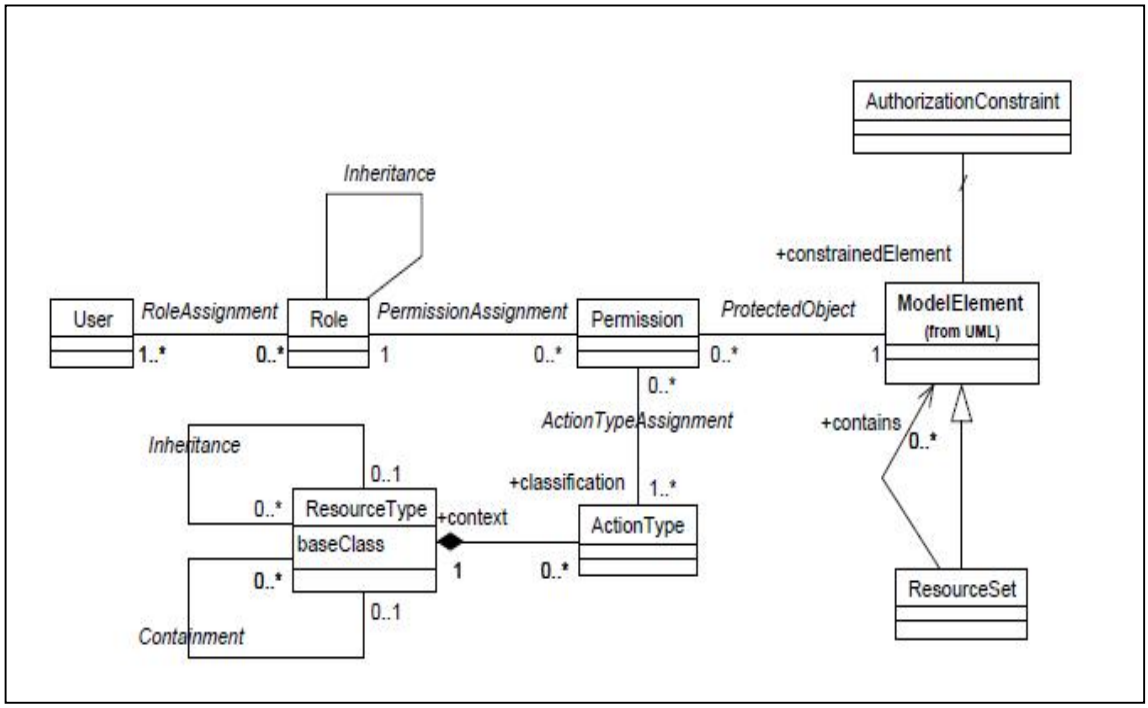

Figure 4.9: SecureUML Metamodel (Adapted from Lodderstedt *et al.,* 2002)

4.7 Summary

The syntax and semantics of the security extension of UML diagrams are discussed by using an example in this chapter. Among the discussed modeling languages, Mal activity diagrams, SecureUML and UMLSec are used at the design stage.

Table 4.1 shows the summary of the concepts of the approaches presented in this chapter. In the first column, *Threat definition*, we have shown how threat (malicious acts) is realized by the specific modeling language. *Phase in software development cycle* shows the level at which the modeling language is used. *UML notations* mean which UML concept and notation is used in the modeling language. In *Mitigation mechanism* column, we have discussed how the security related concepts are addressed in the modeling process. *Tools* column shows whether the modeling language has any tools to generate code from the model or not.

We will further investigate Mal activity diagrams and SecureUML for ISSRM domain model concepts for the alignment with ISSRM domain model in later chapters (Chapter 5 and 6). We choose these two approaches because of the following reasons,

1. As this research is focused on design stage, we select modeling languages that work at the design stage.

2. Mal activity diagrams are selected because it is used from late requirement to design stage. We have already seen concept alignment of requirement engineering languages (e.g., Secure Tropos and Misuse case) in Chapter 3. So, Mal activity diagrams will be more smooth and continuous move between requirement engineering and design stage.

3. We select SecureUML because it is defined with an explicit meta model. It also provides overall security control mechanism for secure information system.

Table 4.1 : A Comparative Study of Security Modeling Language

Name	Threat definition	Phase in Software development cycle	UML notations	Mitigation mechanism	Tools
Abuse case diagrams	By introducing abuse case and textual description	Late requirement stage	Use case	By Introducing Use case but in different diagrams	No
Misuse case diagrams	By introducing misuser, misuse case and textual description.	Late requirement stage	Use case	By Introducing Use case (security use case) in the same diagram	No
Mal activity Diagrams	It uses Mal-swimlane, Mal-activity and Mal-decision	Late requirements to design stage	Activity diagram	Uses Mitigation Activity and MitigationLink	No
UMLSec	By introducing UML profile with stereotypes, tags and	Design stage	The whole UML profile	Uses *UML* profile and tags.	Yes
SecureUML	By combining class with stereotype <<role>> and an association class with stereotype <<permission>>	Design stage	Mainly class diagrams	Uses Authorization Constraint	No (But MS visio template is available[7])

[7] http://www.secureuml.com-about.com/

PART II

II. Security Risk Management for IS Design

This part presents the concept alignment of ISSRM domain model and design level modeling languages. It consists of two chapters.

In the fifth chapter, *Alignment of Mal Activity Diagrams with ISSRM Domain Model,* we discuss about how Mal activity diagrams can support security risk management following ISSRM domain model. We use Mal activity diagrams constructs to discuss an online banking example (Annex A.1) following ISSRM process. We also present the asset modeling, threat modeling and security treatment modeling of the given example by using Mal activity diagrams constructs. The output of the chapter is a concept alignment table (Table 5.1).

In the sixth chapter, *Alignment of SecureUML with ISSRM Domain Model,* We discuss about the concept alignment of SecureUML and ISSRM domain model. Here, we use the same research method and example as Chapter 5. The outcome of this chapter is an alignment table (Table 6.1) between SecureUML and ISSRM domain model.

CHAPTER 5

Alignment of Mal Activity Diagrams with ISSRM Domain Model

In this chapter we will look into the security modeling language at the design stage, more specifically, Mal activity diagrams (Sindre, 2007). In section 4.5, we have introduced Mal activity diagrams. Here we will follow the method described in Section 3.1 to understand how Mal activity diagrams could support security risk management. Our observations will result into a concept alignment between ISSRM concepts and Mal activity diagrams constructs. We will investigate how the concepts of Mal activity diagrams addresses the security risk management issues at the design stage by using a running example (Annex A.1). This will contribute to our third research question: *How could the security modeling language support security risk management at the design stage?*

We will narrow down our running example of online banking system to email correspondence between bank official and client to understand, identify and comprehend the security risk related concepts in detail. We will use ISSRM process (discussed in Section 2.2.3) to identify and investigate security risk management concepts of email correspondence by using the concrete syntax of Mal activity diagrams.

5.1 Modeling with Mal Activity Diagrams

In this section we will analyze how Mal activity diagrams support ISSRM concepts by using the online banking example (Annex A.1). We will follow ISSRM process and use Mal activity diagrams constructs to analyze security risk management concepts of the online banking system example here.

(a) Context and Asset Identification. In Figure 5.1, we present a context of our example- it is shown by activity diagram for online bank. It shows email correspondence between bank employee and the bank client, more specifically the bank employee has sent an email by asking the client to update his home address. In this diagram three actors are presented by *client, bank official* and *online banking servers* (represented as Mal activity diagrams construct `swimlane`).

The bank official is used to *Send email to update the home address* (`Activity`) to start the process. The client r*eceives email* (`Activity`) and then *opens email* (`Activity`). Next, he *loads the website* (`Activity`) in the web browser and browses the login page. In the login page he *puts the username and password* (`Activity`) and *pushes login button*

(Activity). Then the login information (username and password) is sent to the *online banking sever* (Swimlane). The server *validates the user* (Decision), if the account exits in the system then it *redirects the user to his homepage* (Activity). The client then *updates the home address* (Activity) and *logout* (Activity) from the system. If the client's account does not exist or provides wrong username or password then he will be redirected to the login page again.

Bank has a standard procedure to create the account for the client. After creating the login credentials, bank officials send this login information to the client. The client can then login to the system by using these credentials. He can also change his password to ensure that nobody knows his password.

(b) Security Objective Determination. Security objectives are often defined in terms of *confidentiality, integrity* and *availability*. Mal activity diagrams deal with the dynamic behavior of the system and does not support security constraint of the static information (e.g., login information). Here, maintain the integrity of the message sending process can be seen as the security objective.

(c) Risk Analysis and Assessment. In Figure 5.2 we identify malicious activities which are done by *Hacker* (Mal-swimlane). The *Hacker* targets login information of the client. If the *Hacker* can successfully collect the login information of the client he can bypass online bank account creation procedure. That will violate the integrity of the overall login procedure of the system. With this login information hacker can login to the account of the client that will threaten the integrity of the total login system.

The process starts with a malicious activity *Send an email with malware* (Mal-activity). It is sent by a malicious actor *Hacker* (Mal-Swimlane). *The client* is used to *Receives email* and also *open the email*. Then another malicious activity *Silent installation of malware* (Mal-activity) will install the malware in the client machine. After that when client *Puts the username and password*, this malware will send this login information to the hacker. The Hacker will *Receive login information* (Mal-activity). Now with this login information the hacker can login to the bank account and do all kinds of activities that the client is authorized to do. Here the vulnerabilities of the system lie in the email system and in the client machine. Hackers exploit these vulnerabilities and gain access to the client's account.

(d) Risk Treatment. In this step we choose one out of four risk treatment procedures. We are choosing *risk reduction* (taking actions to lessen the probability and negative consequences of the risk) rather than risk avoidance (decision is taken for not to get involved in any risk situation). Here, we are not taking any decision to avoid the situation but trying to improve the security measure to lessen its probability and effect.

(e) Security requirements definition. In Figure 5.3, we *Enable email filtering* (MitigationActivity) which provide safeguard against malicious email. *Setup anti-malware* (MitigationActivity) will provide safe guard against malware installation and *Enable traffic scanner* (MitigationActivity) is used to monitor the message packets that are received and sent from the client's computer. If any malicious traffic is

detected, it will notify the user. *Enable email filtering* is used before (`MitigationLink`) *Receive email, Setup anti-malware* before (`MitigationLink`) *Load the website and Enable traffic scanner* before (`MitigationLink`) *Send the login info.*

(f) Control Selection and Implementation. We can see the *security module* as a security solution for the system which will implement the security requirements.

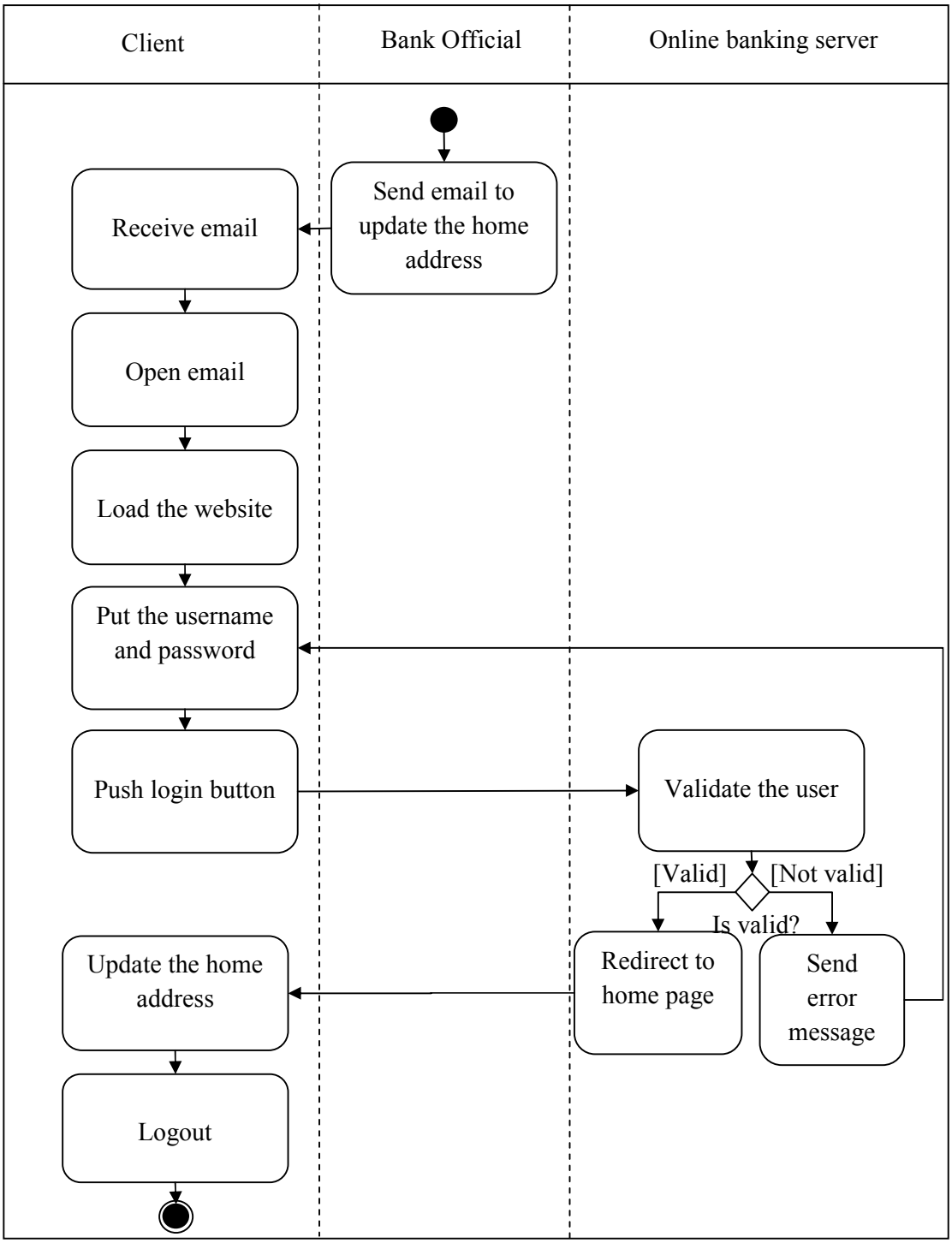

Figure 5.1: Asset modeling: message sending process

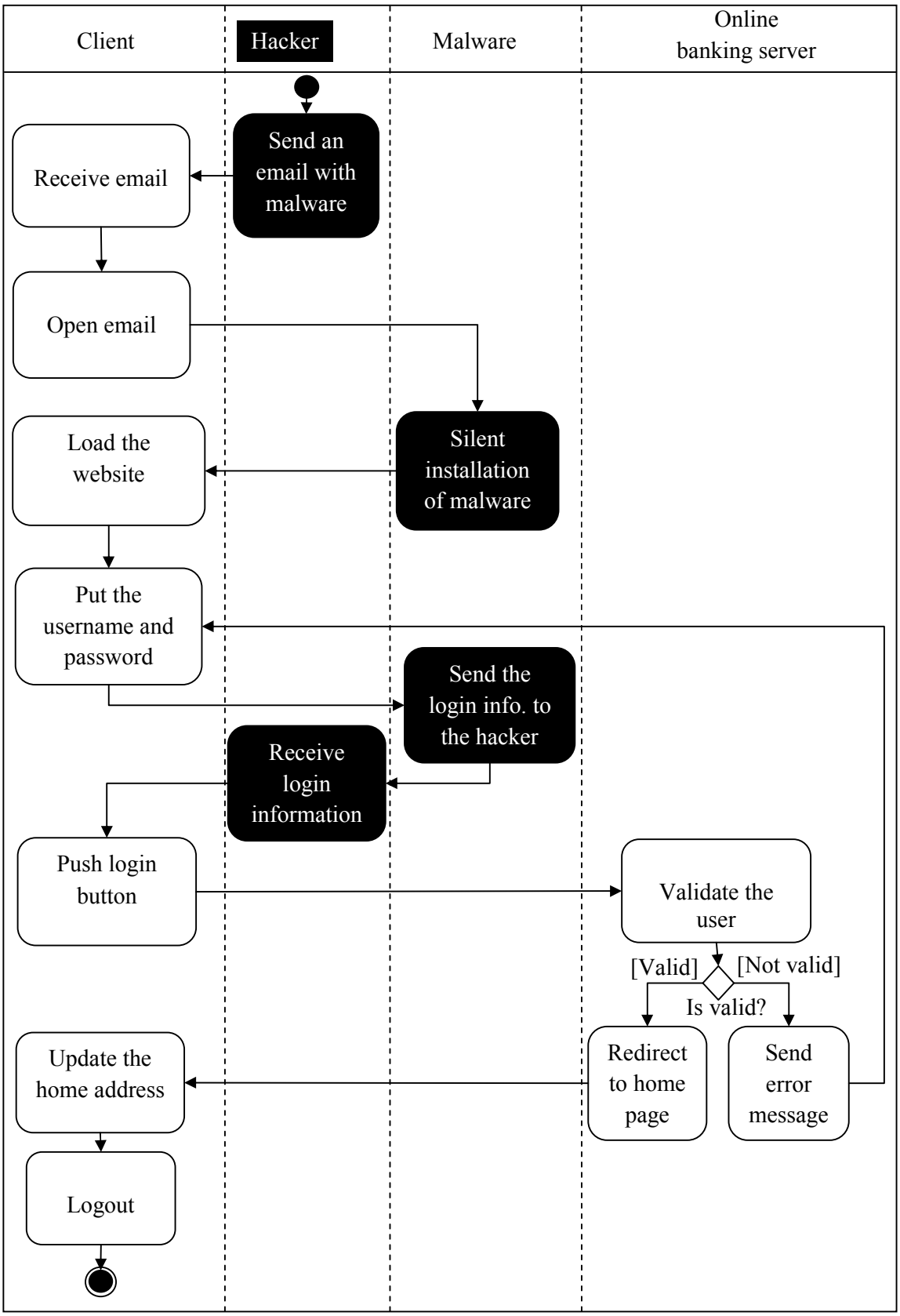

Figure 5.2: Threat Modeling by Mal Activity Diagrams

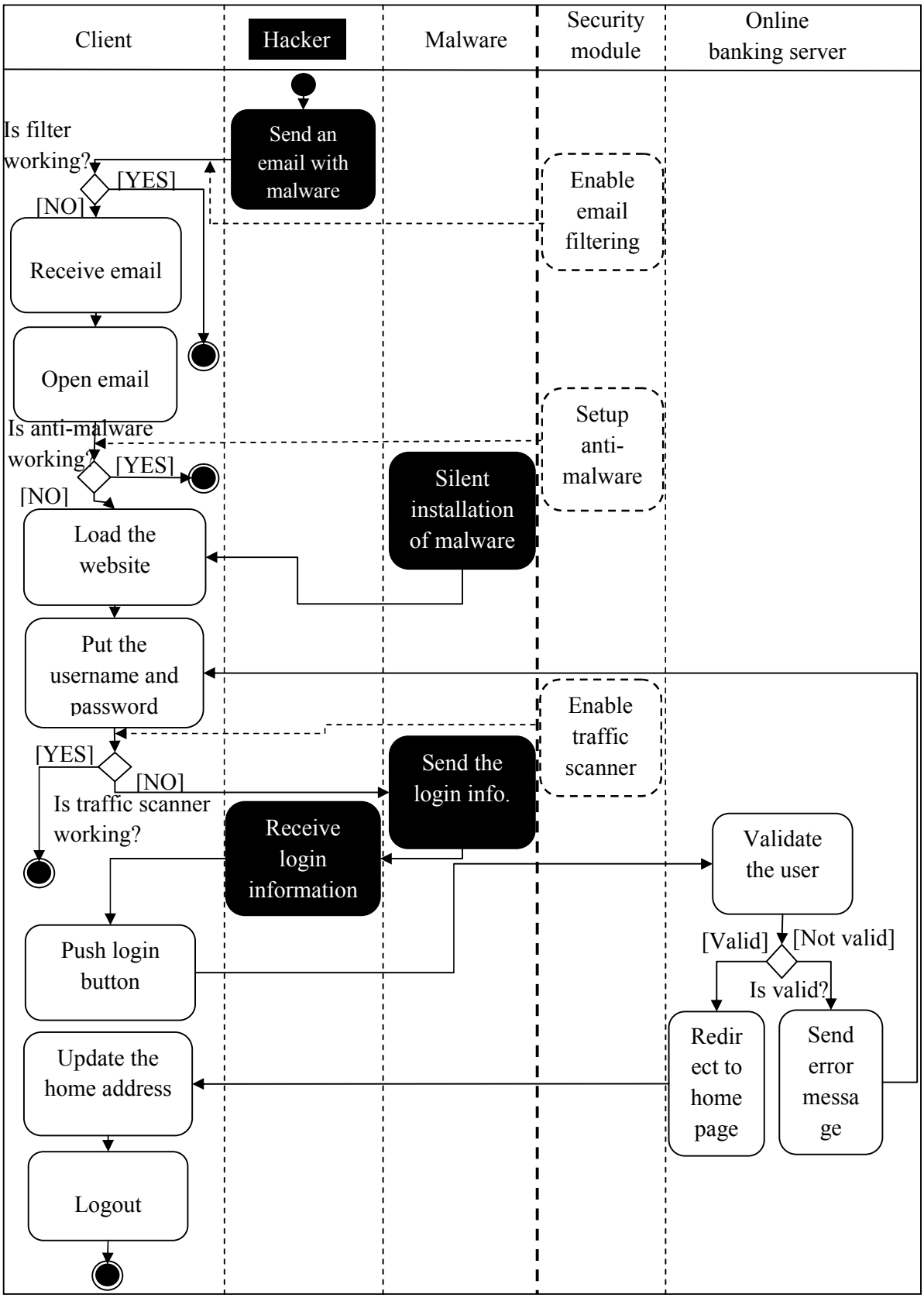

Figure 5.3: Modeling of Security Requirements

5.2 Mal Activity Diagrams and ISSRM Domain Model

(a) Asset-related Concepts. ISSRM *asset* represents anything that has value for the organization and also plays a vital role to achieve the organization's goal. *Business asset* is defined as the information, process, skill that is essential for the business. Activity diagrams are used to show the process in any organization. Thus its constructs: `Activity`, `Decision` and `ControlFlow` are mapped with the *business asset*. *IS asset* is something that supports *business asset.* In Mal activity diagrams, `Swimlane` holds all the constructs and helps to show the sequence of the process with specific actor. `Activity`, `Decision` and `ControlFlow` are also used to support the *business asset*. So, `Activity`, `Decision`, `ControlFlow` and `Swimlane` are aligned with *IS asset*. There is no Mal activity diagrams construct for representing security criterion.

(b) Risk-related Concepts. Malicious actor (*Hacker*) initiates the `Mal-activity`(e.g., *Send an email with malware*) which can be seen as the ISSRM *attack method*. *Attack method* also includes `Mal-Swimlane`, `Mal-decision` and `ControlFlow` which help to execute the `Mal-activity`. *Hacker* (`Mal-Swimlane`) resembles the *threat agent.* A *threat* is represented by the combination of *threat agent* and *attack method. Vulnerability* of the *IS asset* can be identified as no means to scan for incoming email, no controls to check the legibility of installation, no controls for outgoing traffic. *Impact* of the `Mal-activity` on the system can be seen as loss of integrity of the message sending process which will harm to the identified *asset*. Lastly, Mal activity diagrams do not have any visual means to present *risk, vulnerability, event and impact* for the system.

(c) Risk Treatment-related Concepts. The `MitigationActivity` (e.g., *Enable email filtering* in Figure 5.3) can be seen as the countermeasures for a `Mal-activity`. Mal activity diagrams also show in which place `MitigationActivity` should be implemented by `MitigationLink`. Finally, *security module* (`Swimlane`) provides the means to improve the security of the system by implementing the security requirements. So, `Swimlane` can be mapped with the ISSRM security control.

The following table (Table 5.1) shows the concept alignment of Mal activity diagrams and ISSRM domain model. Under the Mal activity diagrams column we write down the graphical constructs of Mal activity diagrams that relate to the ISSRM concepts. In the next column we show the element of example.

5.3 Summary

In this chapter we have shown the concept alignment between ISSRM domain model and Mal activity diagrams. We have discussed the running example (Annex A.1) using Mal activity diagrams constructs and by following ISSRM process. Then we align the constructs with concepts of the ISSRM domain model. The final outcome of this chapter is an alignment table (Table 5.1) which is a semantic alignment between ISSRM domain model concept and Mal activity diagrams constructs.

Table 5.1: Concept Alignment Between Mal activity Diagrams and ISSRM Domain Model

ISSRM domain model		Mal activity diagrams	
		Mal activity diagram	Example
Asset related concepts	Asset	--	--
	Business asset	Activity, Decision, ControlFlow	Send email to update the home address, Receive email, Open email, Load the website , Put the username and password, Push login button, Update the home address, Logout Is filter working? Is anti-malware working? Is traffic scanner working?
	IS asset	Swimlane, Activity, ControlFlow, Decision	Bank official, Online banking server, Validate the user, Redirect to the home page, Send error message, Is valid?
	Security criterion	--	Integrity of the message sending process (Fig. 5.1)
Risk related concepts	Risk	--	--
	Impact	--	Loss of integrity of the message sending process
	Event	--	--
	Vulnerability	--	No means to scan for incoming email, No controls to check the legibility of installation, No controls for outgoing traffic
	Threat	Mal-activity, Mal-Swimlane, Mal-decision, ControlFlow	Send an email with malware, Silent installation of malware, Send the login info. to the hacker, Receive login information Hacker
	Threat agent	Mal-Swimlane	Hacker
	Attack method	Mal-activity, Mal-decision, ControlFlow Mal-Swimlane,	Send an email with malware, Silent installation of malware, Send the login info. to the hacker, Receive login information Malware
Risk treatment related concept	Risk treatment	--	Risk reduction
	Security requirement	MitigationActivity, MitigationLink	Enable email filtering, Setup anti-malware, Enable traffic scanner

	Control	Swimlane	Security module

This concept alignment has shown several limitations of Mal activity diagrams to investigate security risk management at the design stage of the information system development process. One of the main drawbacks is that it cannot present static security criteria (e.g. confidentiality of login information). We suggest couple of improvements for Mal activity diagrams, in the context of security risk management:

- Mal activity diagrams (Sindre, 2007) do not provide guide line how to use its construct. This sometimes may create confusion when we wants to use it in ISSRM process. For example, `Activity` is used for both *business asset* and *IS asset.* One possible solution is to use different color like `irregular activities` (irregular activities performed in good faith (grey)(Sindre, 2007)).
- There is no published (e.g., in conference or journal) work available on how to use (process definition) Mal activity diagrams. So, a process definition(e.g., 5 steps process for Abuse case diagrams (Mayer, 2009)) should be prepared to use it correctly and efficiently.
- No published meta model is available for Mal activity diagrams. We have proposed one in section 4.4 (Figure 4.5 and 4.6). This will help to understand the constructs and their relationship.
- Mal activity diagrams could be improved with additional construct to better cover the concepts of ISSRM. In table 5.1, we see that Mal activity diagrams does not have construct to represent some ISSRM domain model concepts (e.g., *security criterion, risk, impact, vulnerability, risk treatment*). One possible solution is to propose new construct or to provide methodological guideline how to use the existing concepts in the diagram.

CHAPTER 6

Alignment of SecureUML with ISSRM Domain Model

In this chapter we will analyze how SecureUML (Lodderstedt, 2002) supports security risk management. We have introduced SecureUML constructs, meta model and related concepts in Section 4.6. Like the previous chapter, we will follow the same method (in Section 3.1) to map SecureUML to the ISSRM domain model. We will base our analysis on the online banking scenario (Annex A.1). The outcome of this chapter will be a semantic alignment table between SecureUML and ISSRM domain model. It will help the developers to realize security risk management concepts in terms of SecureUML constructs at the design stage. This will also contribute to our third research question: *How could the security modeling language support security risk management at the design stage?*

6.1 Modeling with SecureUML

Here, we will follow ISSRM process (in Section 2.2.3) and SecureUML constructs to analyze the our running example (Annex A.1).

(a) Context and Asset Identification. In Figure 6.1 we illustrate the context of our example. We assume that there are two roles: *Client* and *Bank* (represented as a class with a stereotype `<<Role>>`) are used for Bank account control management. The *client* can access *BankAccount* (a class with a stereotype `<<ModelElement>>`) and transfer money by using its operations (e.g., deposit(), withdraw()). The attribute (e.g., balance) of the *Bank account* shows the current balance of the account. The authorization on *BankAccount* for *Client* is assigned by *ClientPermission* (an association class with a stereotype `<<Permission>>`). The *client* can *read* and *update* `(ActionType)` the *BankAccount* information. Like the client role, Bank officer role is also authorized to the account by `Permission` (*BankOfficerPermission*). Bank officer can perform *read, create* and *delete* `(ActionType)` operation on *BankAccount*. One client can have more than one accounts and one account can be managed by more than one Bank officers. Bank officer also can manage many accounts.

(b) Security Objective Determination. Security objectives are often defined in terms of *confidentiality, integrity and availability*. Here, we are concerned about the manipulation of *balance* information. Thus, we focus on the *integrity* of the *balance* (an attribute of a class with a stereotype `<<ModelElement>>`) information.

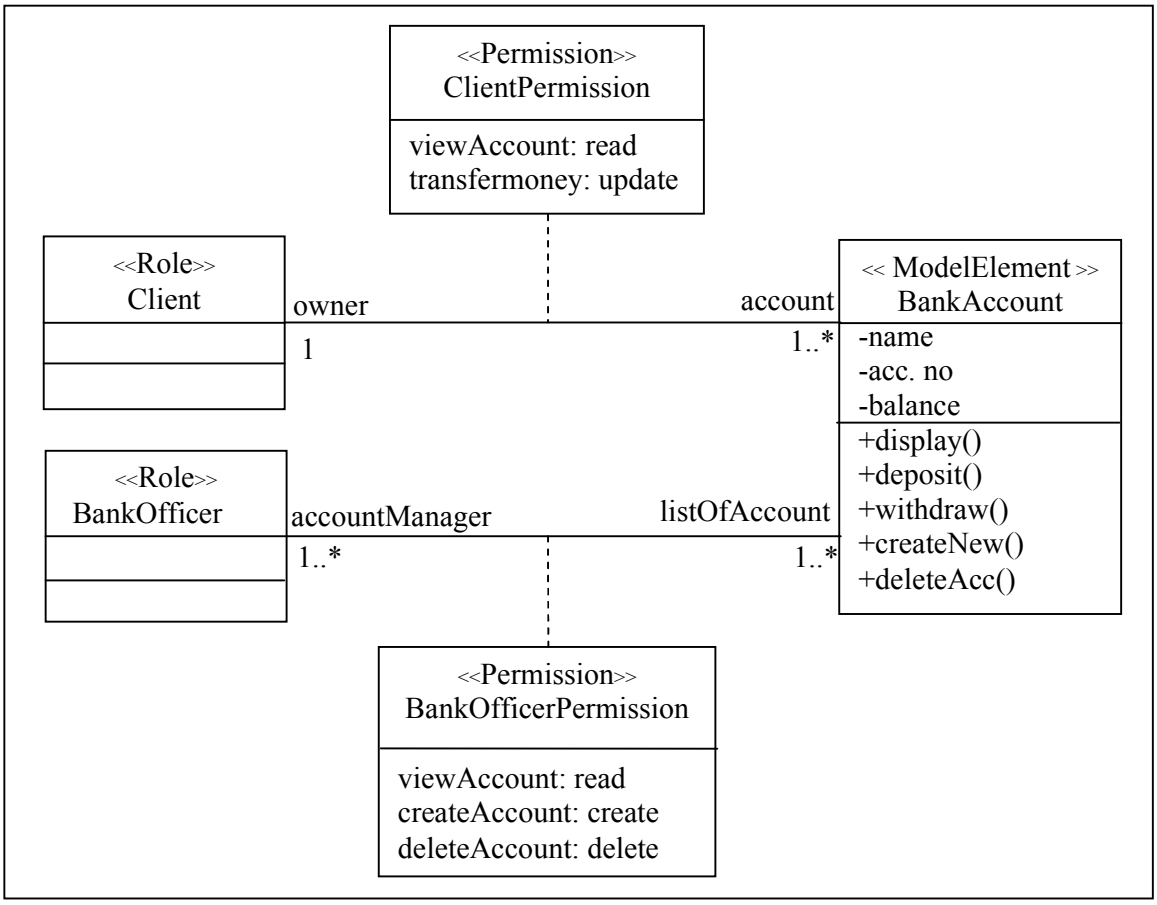

Figure 6.1: Asset Modeling by SecureUML

(c) Risk Analysis and Assessment. In Chapter 5, we have seen that malicious actor (*Hacker* represented as a class with a stereotype `<<Role>>`) can steal the login information by using email system and malware (shown in Figure 5.2). Here we are assuming that the hacker has already received the login information. When the hacker will login to the system with the stolen login information he will get the same access level like *Client* (Figure 6.2). Now, he can execute all kinds of operation (e.g., *illegalTransfer: update* represented as attribute an association class with a stereotype `<<Permission>>`) on the *Bank account* as *Client*.

(d) Risk Treatment. In this step we choose one out of four risk treatment procedures. We are choosing *risk reduction* (taking actions to lessen the probability and negative consequences of the risk) rather than risk avoidance (decision is taken for not to get involved in any risk situation). Here we are not taking any decision to avoid the situation but trying to improve the security measure to lessen its probability and effect.

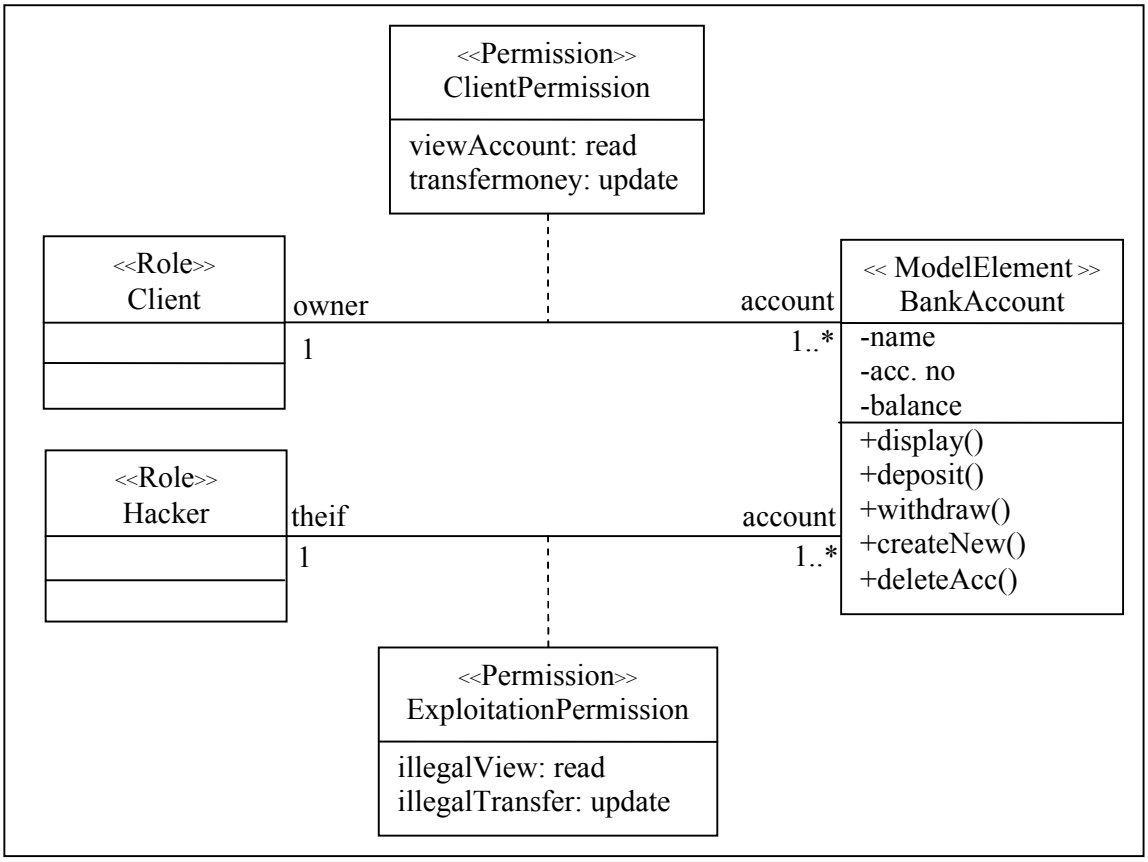

Figure 6.2: Threat Modeling by SecureUML

(e) Security Requirements Definition. By setting `AuthorizationConstraint` we can put our security requirement (Figure 6.3). Here, constraint (*OneTimePassword)* will ensure that even if the login information is compromised the malicious actor (*Hacker*) cannot transfer (*transfermoney:update* attribute of an association class with a stereotype `<<Permission>>`) money to other account.

To ensure this `AuthorizationConstraint` to work every client should be given a printed one time code book. They will use this code as the confirmation of any *create, update* or *delete* operation on *Bank account*. For example, this code is used to confirm the money transfer. The main characteristic of this code is that it can be used only once and after that it becomes invalid. The banking system check this code with the code book for that specific client. So, even if the *hacker* steal the login information he can only see (*illegalView: read*) the account information. If he tries to transfer money he will be prompted for the one time code as confirmation. As he does not have access to this one time code book he could not transfer the money to his account.

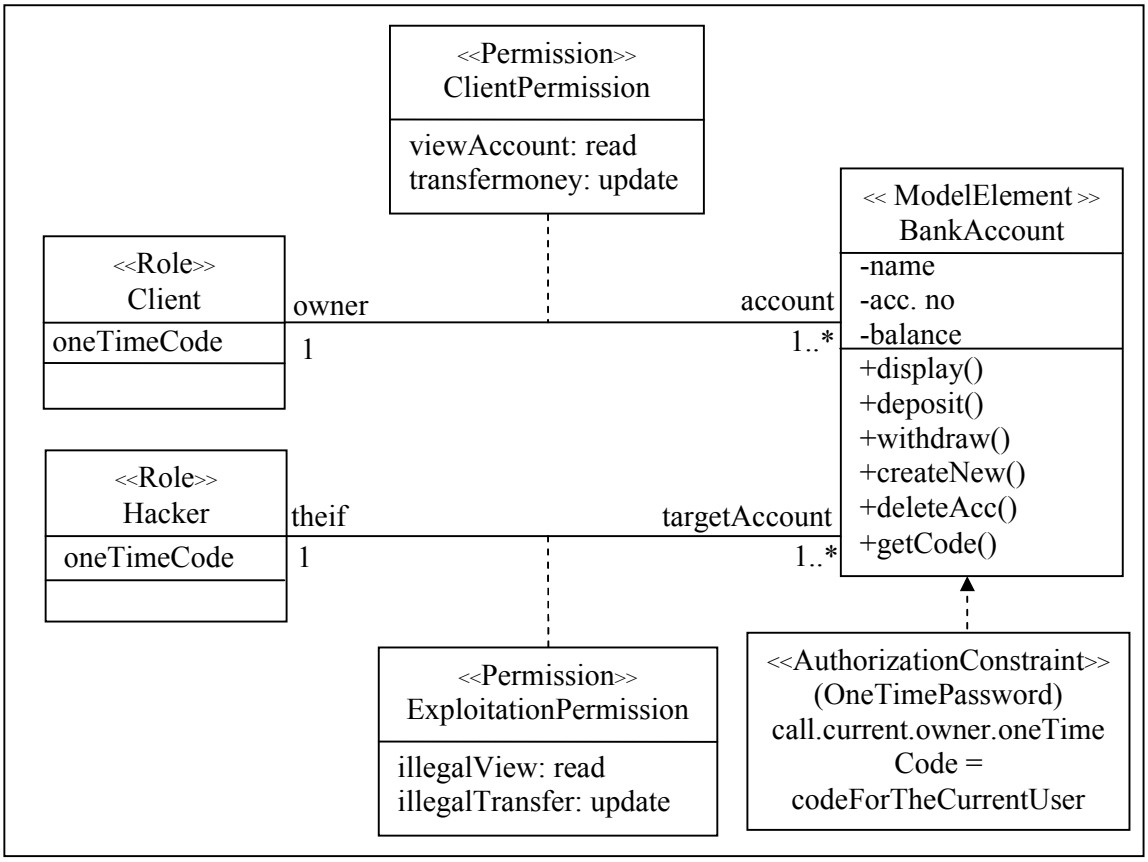

Figure 6.3: Modeling of Security Requirements

(f) Control Selection and Implementation.
From this model we can transform to implementation (Matulevičius, 2011). The overall model defines rather a control mechanism and links it to the secured assets than separate concepts of ISSRM.

6.2 SecureUML and ISSRM Domain Model

(a) Asset-related Concepts. ISSRM *asset* is anything that is valuable and vital for achieving the origination's goal. B*usiness asset* is defined as the information, process, plan and skill essential to the business. In SecureUML, a class with a stereotype `<<ModelElement>>` (*BankAccount*) corresponds to protected resources. Protected resources represent the module that is valuable for online banking system to provide services. Thus a class with a stereotype `<<ModelElement>>` can be mapped with *asset*. We align *BankAccount* with ISSRM *asset* rather than *business asset* because all the information in this class are not critical (e.g., *name* or *acc. no*) for online banking system but needed to provide the service. The attribute (*balance*) of the class with a stereotype `<<ModelElement>>` represents the information that is critical for the client as well as for banking system. Thus, it is mapped with *business asset.*

A class with a stereotype `<<Role>>` provides the way to use this system. An association class with a stereotype `<<Permission>>` supports the user with proper authorization to execute an operation on one or more protected objects or resources. The operation (e.g., *withdraw()*, *deposit()*) of the *Bank account* is the means which helps to work on the balance information. In this perspective, these constructs (Role, Permission, Operation of ModelElement) can be

aligned with *IS asset.* Though SecureUML is focused on security (access control) but it does not have any construct to represent ISSRM *security criteria.*

(b) Risk-related Concepts. In principal, SecureUML does not have the support to represent security risk. **"**at the methodological level SecureUML only focuses on the solution domain.**"** (Matulevičius, 2010).

However, we have used SecureUML to define the threat model (in Figure 6.2) of the running example to show the negative/harmful effect. Here, hacker (a class with a stereotype `<<Role>>`) who takes the unlawful benefits by transferring money illegally (*illegalTransfer: update*) can be aligned with *threat agent.* The means used by the *threat agent* is *attack method.* Thus, the *attack method* (*illegalTransfer: update*) can be aligned with the attribute of an association class with a stereotype `<<Permission>>`. Hacker uses this permission to exploit the vulnerability and targets the *IS assets*. *Vulnerability* of the *IS asset* can be defined implicitly (no security confirmation mechanism before operation is performed). Exploitation of this vulnerability will result into an *impact* (Loss of integrity of the money transfer) which will cause harm to the *asset*. We can implicitly identify the *vulnerability* and *impact* but SecureUML does not have any construct to represent them. SecureUML also does not have any construct to define *risk* and *event.*

(c) Risk Treatment-related Concepts. It deals with the risk treatment decision and implementation of security requirements in order to mitigate *risk*. Here we have selected *risk reduction* as a risk treatment decision for our example. ISSRM security requirement is represented by `AuthorizationConstraint` (*OneTimePassword*). This constraint provides the counter measures against illegal money transfer. The overall SecureUML approach implements the security requirement. Thus, it can be aligned with control.

Table 6.1 shows the concept alignment of SecureUML and ISSRM concepts. Under the SecureUML column we write down the graphical constructs of SecureUML that relates to the corresponding ISSRM concepts and in the next column we have shown the example.

6.3 Summary

In this chapter, we have shown the concept alignment between ISSRM domain model and SecureUML. We have analyzed the concepts by using a running example. The final outcome of this chapter is an alignment table (Table 6.1). Our analysis shows that SecureUML has several limitations to define the security risk management concepts. Here we will suggest few improvement for SecurUML from security risk management perspective.

- Several concepts (e.g., *risk, event*) do not have any construct to represent. The solution could be to introduce new stereotypes or tagged values within existing stereotypes.
- Sometimes same constructs (e.g., `<<Role>>`, `<<Permission>>`) is represented for different ISSRM concepts (e.g., *IS asset* and *attack method*). Specific *tagged value* could be used to differentiate the same constructs (Stereotype).

Table 6.1: Concept Alignment Between SecureUML and ISSRM Domain Model

ISSRM domain model		SecureUML	
		SecureUML	Example
Asset related concept	Asset	Class with a stereotype <<ModelElement>>	Bank account
	Business asset	Attributes of the class with a stereotype <<ModelElement>>	balance, name, acc. no
	IS asset	Class with a stereotype <<Role>>, Association class with a stereotype <<Permission>>, Operations of the class with a stereotype <<ModelElement>>	Client, Bank Officer Client permission, Bank officer permission display(), deposit(), withdraw(),viewAccount: read, createAccount: create, deleteAccount: delete, transfermoney: update
	Security criterion	--	Integrity of the balance
Risk related concept	Risk	--	--
	Impact	--	Loss of integrity of the money transfer
	Event	--	--
	Vulnerability	--	No security confirmation mechanism before operation is performed
	Threat	Class with a stereotype <<Role>>, Association class with a stereotype <<Permission>>	Transfer money by malicious actor
	Threat agent	Class with a stereotype <<Role>>	Hacker
	Attack method	Attributes of an association class with a stereotype <<Permission>>	illegalView: read illegalTransfer: update
Risk treatment related concept	Risk treatment	--	Risk reduction
	Security requirement	Authorization constraint	OneTimePassword
	Control	The whole model as control	Bank account control management

Part III

IV. Validation

This part presents validation of our alignment.

In Chapter 7, we perform a case study by using Mal activity diagrams and SecureUML. We use two different but sequel scenarios from the case study. We identify the key instances of the case study and investigate how they are represented by Mal activity diagrams and SecureUML following ISSRM process. We also find out ISSRM domain model concepts which best fit for these elements. Thus, we have got two alignment tables (Table 7.1 and Table 7.2). We check alignment of these two tables with Table 5.1 and Table 6.1 to validate the concept alignment. This investigation validates our proposed alignment in Chapter 5 and Chapter 6. Finally, we discuss threat to validity to show the limitation of the validation process.

CHAPTER 7

Validation

In this chapter, we will validate the concept alignment of security modeling languages(e.g., Mal activity diagrams and SecureUML) with ISSRM domain model. We will use a case study (described in Section 7.2) about banking information which is exposed to social engineering(Mitnick, 2002) attack to validate our alignment (Table 5.1 and Table 6.1).

7.1 Method of Validation

We will analyze and identify the components of the case study from the security risk management perspective. We will model the case study by Mal activity diagrams and SecureUML. Then we will find out what construct of modeling language is used to represent the element of the case study and to which ISSRM concept the element is matched. Now we take each element and will see the alignment between the construct of the modeling language and ISSRM domain model for this element. Then we will look into our proposed alignment (Table 5.1 for Mal activity diagrams and 6.1 for SecureUML) table to verify that whether the same construct of the modeling language matches with same ISSRM concepts for both the case or not. If we find match for all the elements then it will validate our proposed alignment.

7.2 The Credit Chex Case Study

When someone goes to the bank to open a bank account, the bank uses some third parties to verify his personal information and financial status. Credit chex is such kind of third party organization which provides information to the banks.

Whenever the bank needs the verification, they call credit chex and credit chex replies with the information. To improve the service quality credit chex conducts survey about the service and customer satisfaction from time to time. For the survey they call the banks and discuss about the survey questions and get the response.

The survey is done through telephone. The bank official does not check the phone number of the caller when they answer to the survey questions. This may lead to a social engineering attack. The attacker may have collect secret information which he can use to disguise himself as a bank official to the credit chex office and collect client's personal and financial information. A social engineering method on this case study is presented in (Mitnick, 2002). We will analyze it using Mal activity diagrams and SecureUML in the following sections.

7.3 Managing Risks Using Mal Activity Diagrams

In this survey procedure there are two users: bank officials and credit chex official.

Bank Officials. Bank official knows the banking information that the credit chex official wants to know in the survey. The credit chex official asks him for a survey. If he agrees to

participate then the surveyor will start asking questions. During the survey the bank official can skip any question.

Credit Chex Official. He is used to ask the bank official for the survey. He can be accepted or rejected by the bank official. If the bank official agrees to answer his survey questions he will ask the questions. He also writes down the responses of the questions.

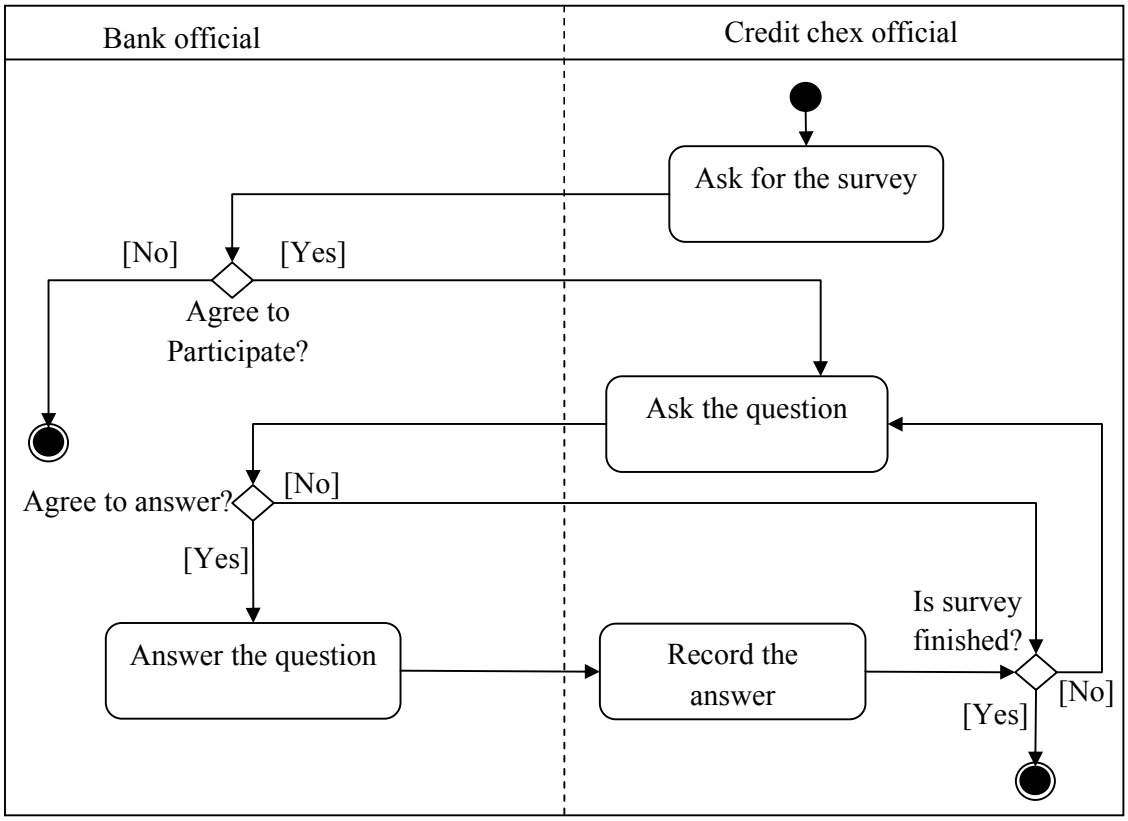

Figure 7.1: Modeling of Survey Process by Mal activity diagrams

Figure 7.1 shows a survey scenario. The survey starts by asking (*Ask for the survey* represented as `activity` in Mal activity diagrams) the banking official for the survey. Bank official can *Agree to Participate* (`Decision`) or not. If he agrees to answer the survey question, the credit chex official will *Ask the question* (`activity`). If the bank official *Agrees to answer* (`Decision`) then the credit chex official will *Ask the question* (`activity`) and he will *Answer the question* (`activity`). The check chex official will then *Record the answer*(`activity`). The credit chex official check whether the survey is finished or not (*Is survey finished?)* after recording the answer. He will also check if the bank official skips any question (*Agree to answer?* as `Decision`) or not. If the survey is not finished (Is survey finished? as `Decision`) then the surveyor will ask the next question.

In Figure 7.2, we can see how an *attacker* (`Mal-swimlane`) can reveal the secret information (e.g., merchant-id) by manipulating the survey procedure. In this case he will disguise himself and act like a legitimate credit chex official to the bank official. In this scenario, he breaks the integrity of the survey and *Asks mal-questions* (`Mal-activity`).

This mal-questions (e.g., What is your bank's Merchant-ID?) will reveal the secret information. Usually, when (*Time to ask mal-question?* Represented as `Mal-decision`) the attacker will gain the confidence of the bank official, he will ask this mal question.

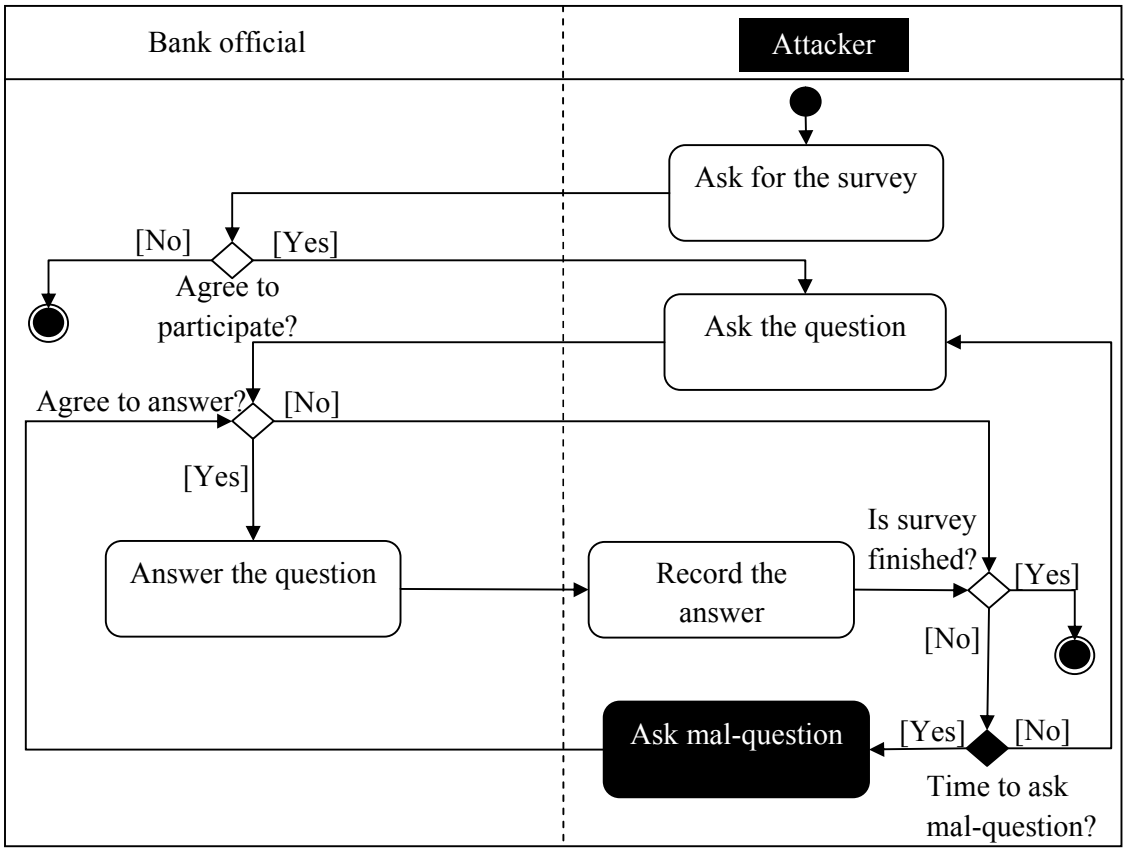

Figure 7.2: Social Engineering Attack on Survey Process

If the bank officials are not security aware he may find this question (e.g., *What is your bank's Merchant-ID?*) as harmless because he used to use this information publicly with his colleague as well as with credit chex officials. If the malicious user gets this information he can later use this information. He may disguise himself as a bank official to the credit chex office and ask any information about any specific client. Later he may use this information for unlawful activities (e.g., identity theft) or sell it to others.

The security mechanism should be in place to avoid this kind of disclosure. Figure 7.3 shows the security mechanism how to protect merchant-id. The bank can store the credit chex phone number that they use for conversation. Then the bank official can *Check the phone number* (`MitigationActivity`) before (`MitigationLink`) they reply to the survey questions. It is not a full proof solution because sometimes malicious user can tamper the phone system and steal the caller id. The bank officials should be trained (Check legitimacy of question as `MitigationActivity`) about this kind of situation. So that even if the malicious actor manages to tamper the phone he could not reveal the secret information from the bank employee.

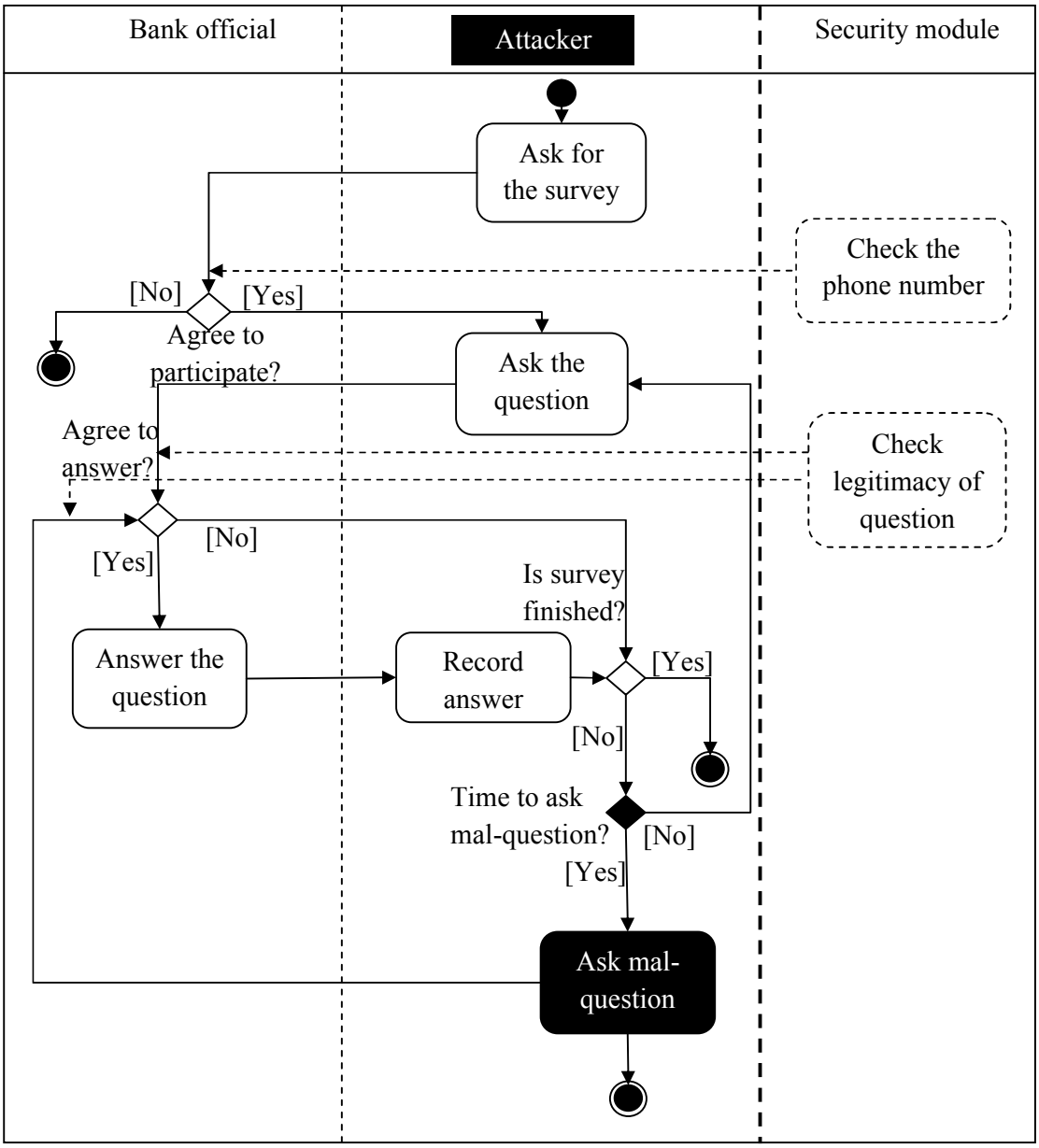

Figure 7.3: Security Requirements to Protect the Integrity of the Survey Process

7.4 Managing Risks Using SecureUML

We are assuming that the attacker has gained the merchant-id by a successful attack (presented in Figure 7.2). Here we will focus on how the attacker uses this secret information to gain access to client's personal and financial information (a class with a stereotype `<<ModelElement>>`) (Figure 7.4).

In SecureUML a bank employee use the position of a *BankOfficial* (represented as a class with a stereotype `<<role>>`) to get access to the client's information .

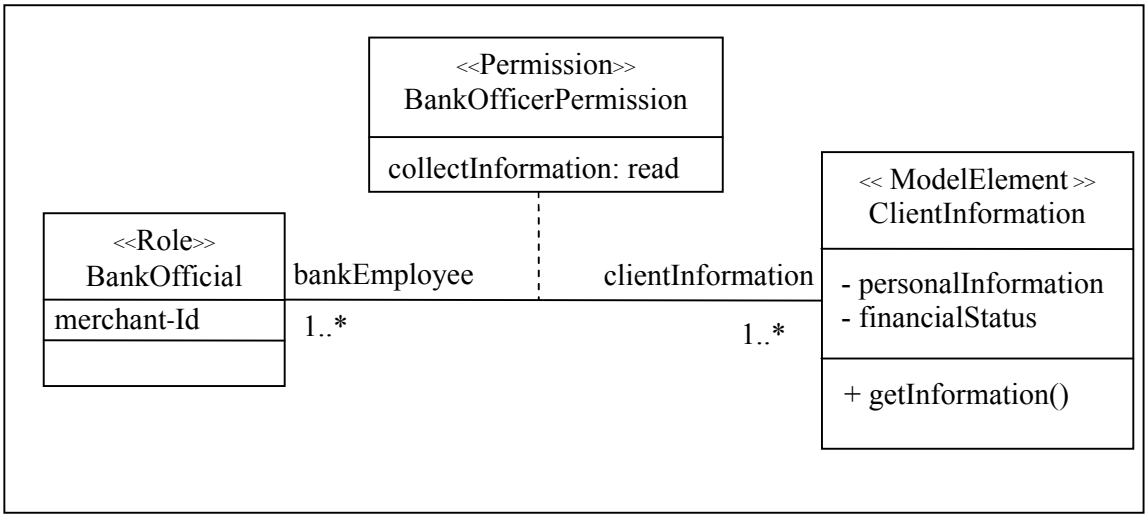

Figure 7.4: Modeling Client's Information System by SecureUML

The authorization is given to bank employee by *BankOfficerPermission* (an association class with a stereotype `<<Permission>>`). The bank employee may ask and get (*collectInformation: read* as an attribute of an association class with a stereotype `<<Permission>>`) these information from the credit chex office.

The bank official is interested about the *personalInformatio*n and *financialStatus* (both represented as attribute of the class with a stereotype <<ModelElement>>) from the ClientInformation system. He accesses this vital information by using *getInformation*() method (represented as operation of the class with a stereotype <<ModelElement>>). Whenever the bank employee wants to access this information he has to provide *merchant-Id* (represented as an attribute of the class with a stereotype <<Role>>) of the bank for authentication.

Figure 7.5 shows what information an attacker can get if he gains access to client's information. Here, the attacker disguises himself as bank official (pre-condition: successful attack described in Figure 7.2) and gains the permission (*ExploitationPermission* as an association class with a stereotype `<<Permission>>`) on the client's information.

This is a violation of client's privacy and the confidentiality of the information is revealed to unauthorized person. The attacker can now use this information for other illegal activities (e.g., identity theft) or also can sell this information to third parties.

BankPhoneNumber `(AuthorizationConstraint)` is used to put constraint on the (protected)information in SecureUML (in Figure 7.6). Here, we use some constraint to mitigate the risk identified in Figure 7.5. Like the bank (discussed in section 7.3), the credit chex office should keep the record of the telephone number that the client (Bank) uses to call for any information. If the credit chex official checks the telephone number before providing any information to the caller, it will reduce the risk.

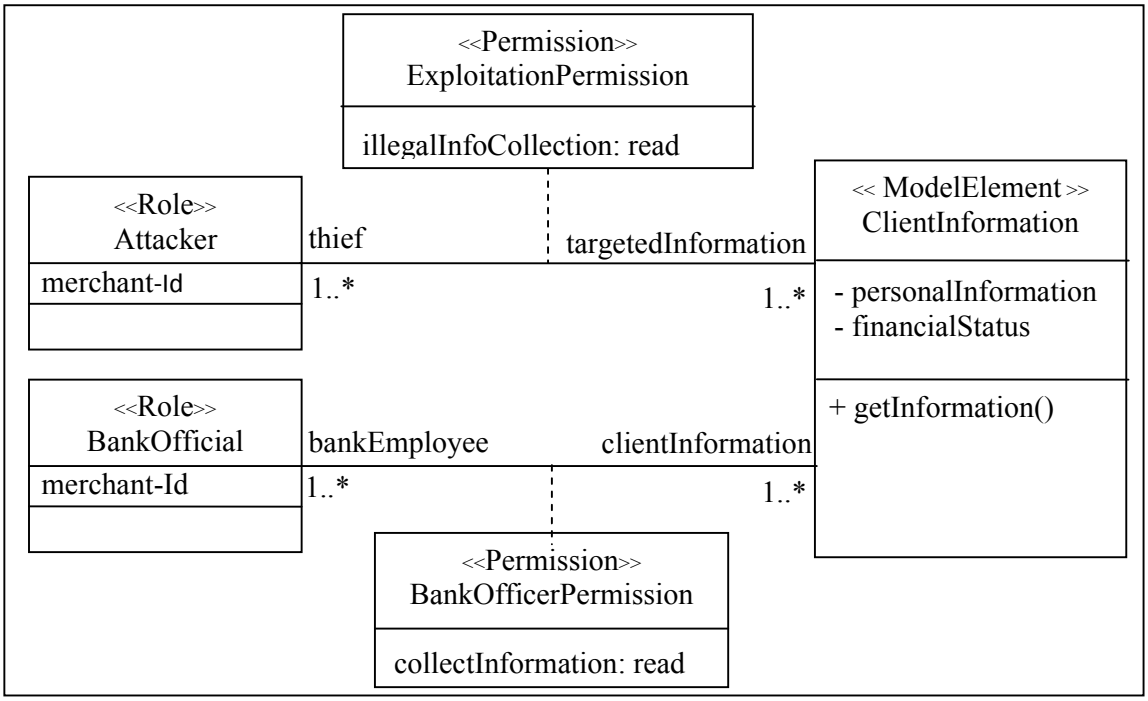

Figure 7.5: An attacker's Access Level to the Client's Information System

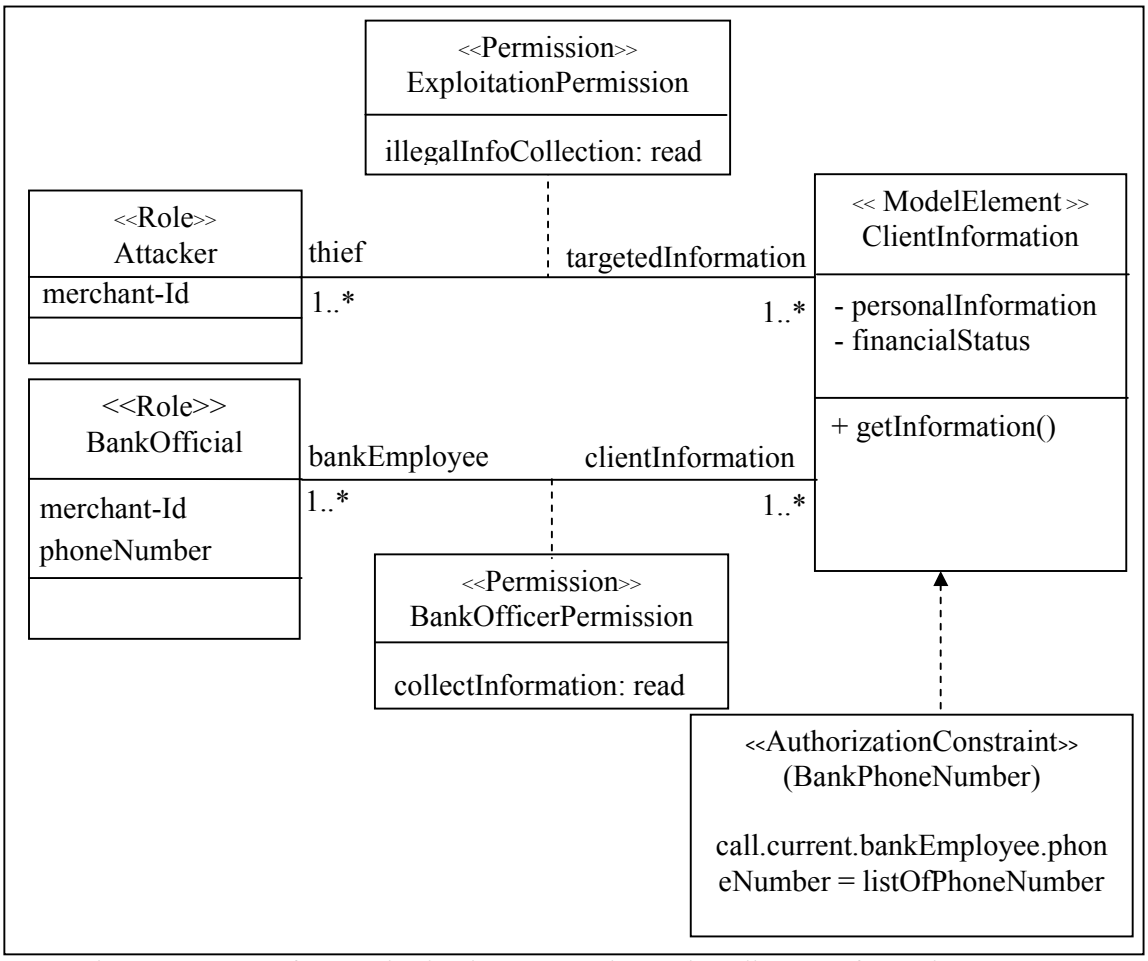

Figure 7.6: Use of an Authorization Constraint to the Client's Information System

7.5 Observation

In Section 7.3 , we have used Mal activity diagrams to present the survey procedure and in Section 7.4, we have used SecureUML to present the correspondence between bank official and the credit chex official for client's information. Now we will use the validation method discussed in Section 7.1 to validate our work.

7.5.1 A Case of Mal Activity Diagrams

In this Section we will discuss about the elements of the case study as presented in Figure 7.1, 7.2 and 7.3 and analyze them. The first element "*Ask for the survey*" is represented as an `activity` in the diagram (Figure 7.1). This element is a vital part of the survey process. So, from this perspective it matches with the ISSRM *business asset*. Thus, `activity` is aligned with ISSRM *business asset* for this element. If we get back to Table 5.1, we see that `activity` is aligned with ISSRM *business asset*. This validates our alignment of `activity` with *business asset* in Table 5.1.

Now we will look into another element of the model "*Agree to participate?*". This is a choice (represented as `decision` in Figure 7.1) taken by the bank official. If he agrees then the survey process will continue otherwise the process will terminate there. So, it is a vital business decision for the survey procedure. Thus, this element is aligned with ISSRM *business asset*. If we look into Table 5.1, we see that `decision` is also aligned with *business asset* there. This also validates our alignment in Table 5.1.

In the survey process "*Bank official*" (represented as `Swimlane` in Figure 7.1) is playing supporting role by providing information to the credit chex official. From this perspective bank official can be aligned with *IS asset*. If we see in Table 5.1, we find that `Swimlane` is aligned with *IS asset*. This also validates our alignment in Table 5.1.

The "*Attacker"* (represented as `Mal-swimlane` in Figure 7.2) disguises himself as a credit chex official to the bank and reveals the secret information. Thus it resembles with ISSRM *threat agent*. `Mal-swimlane` is also aligned with *threat agent* in Table 5.1. Therefore, this validates our alignment in Table 5.1.

An attacker is used to "*Ask mal-question*" (represented as `Mal-activity` in Figure 7.2) in order to disclose the secret information (e.g., merchant-id). This activity helps him to gain unauthorized information. From this perspective, it is aligned with ISSRM *attack method*. In Table 5.1, `Mal-activity` is also aligned with *attack method*. Thus, it validates our alignment in Table 5.1.

By using "*Time to ask mal-question?*" (represented as `Mal-decision` in Figure 7.2) the attacker choose the right time for asking the mal-question. As this decision is related to a harmful action so it is aligned with ISSRM *action method*. In Table 5.1, `Mal-decision` is also aligned with *attack method*. Thus, it validates our alignment in Table 5.1.

The element "*Check the phone number*" (represented as `MitigationActivity` in Figure 7.3) is used to mitigate the *risk* described in Figure 7.2. Thus it is matched with ISSRM

security requirement concept. In Table 5.1, `MitigationActivity` is also aligned with *security requirement*. Thus, it validates our alignment in Table 5.1.

The "Security module" (represented as `Swimlane` in Figure 7.3) provides the means to implement the security requirements to countermeasure the potential risk. So, it is aligned with ISSRM *control*. If we look at Table 5.1, we see that `Swimlane` is also aligned with ISSRM *control*. This also validates our proposed alignment in Table 5.1.

In Table 7.1, we have presented the concept alignment between Mal activity diagrams and ISSRM domain model focusing on the elements of the case study.

Table 7.1: Concept Alignment Between Mal activity Diagrams and ISSRM Domain Model

Serial No.	Element of the case study	Mal activity diagrams	ISSRM domain model
1.	Ask for the survey	Activity [Ask for the survey]	Business asset
2.	Agree to participate?	Decision [Agree to participate?]	Business asset
3.	Bank official	Swimlane [Bank official]	IS asset
4.	Attacker	Mal-swimlane [Attacker]	Threat agent
5.	Ask mal-question	Mal-activity [Ask mal-question]	Attack method
6.	Time to ask mal-question?	Mal-decision [Time to ask mal-question?]	Attack method
7.	Check the phone number	MitigationActivity [Check the phone number]	Security requirement
8.	Security module	Swimlane [Security module]	Control

7.5.2 A Case of SecureUML

In this Section we will discuss about the scenario presented by SecureUML in Figure 7.4, 7.5 and 7.6.

In Figure 7.4, we see how a bank official accesses the "*ClientInformation*" (a class with a stereotype `<<ModelElement>>`) from the credit chex office. Client's information is the vital information for the system. Thus it resembles to ISSRM *asset*. If we see the alignment in Table 6.1, we see that a class with a stereotype `<< ModelElement >>` is aligned with *asset.* So, the mapping here validates our alignment in Table 6.1.

The critical information for the business are "*personalInformatio*n" and *"financialStatus"* (both are attribute of the class with a stereotype `<<ModelElement>>`). Thus it aligns with the ISSRM *business asset*. In Table 6.1, an attribute of the class with a stereotype `<<ModelElement>>` is also aligned with *business asset*. Thus, this alignment here validates our proposed alignment in Table 6.1.

The bank employee gets the authorization on client's information by "*BankOfficerPermission*" (an association class with a stereotype `<<Permission>>`). This permission enables the bank official to access client's information, so it is aligned with

ISSRM *IS asset.* An association class with a stereotype <<Permission>> is also aligned with *IS asset* in Table 6.1. Therefore, the alignment here validates our alignment in Table 6.1.

The bank employee gets access to the client's information by using "*getInformation()*" method (operation of the class with a stereotype <<ModelElement>>). Since it helps to get the desired information it maps with ISSRM *IS asset*. In Table 6.1, we see that operation of the class with a stereotype <<ModelElement>> is aligned with *IS asset*. Thus, this validates our alignment in Table 6.1.

If an "*Attacker*" (a class with a stereotype <<Role>>) gains access to the system, he can reveal the secret information and can use it for illegal activities (e.g., identity theft) or sell it to other parties. Thus it can be mapped with ISSRM *threat agent*. In Table 6.1, a class with a stereotype <<Role>> is also aligned with *threat agent*. Therefore, it validates our alignment in Table 6.1.

The attacker uses "*illegalInfoCollection: read*" (an association class with a stereotype <<Permission>>) to gain illegal access to the client's information. So, it can be mapped with ISSRM *attack method*. In Table 6.1, an attribute of an association class with a stereotype <<Permission>> is also aligned with *attack method*. Thus, this validates our alignment in Table 6.1.

"*BankPhoneNumber*" (AuthorizationContraint) is used to mitigate the risk (presented in Figure 7.5) and can be aligned with ISSRM *security requirement*. In Table 6.1, AuthorizationConstraint is also aligned with *security requirement*. Therefore, it validates our alignment in Table 6.1.

The whole system is designed by using role based access control which implements mitigation procedure to protect client's information. So, from this perspective the whole SecureUML approach is aligned with ISSRM *control.* In Table 6.1, the Whole SecureUML model is also aligned with *control*. Thus, the alignment here validates our alignment in Table 6.1.

In Table 7.2, we have presented the concept alignment between SecureUML and ISSRM domain model focusing on the elements of the case study.

7.6 Threat to Validity

The ideal case study is quite impossible to reach. A case study always suffers of limitations – also called threats to its validity. General limitations of case studies are, for example:

- Reduction of the domain of the IS in order to fit the scope of the problem to treat;
- Increased focus on positive results and vague discussion of negative outcomes.

In Credit Chex case study, we have modeled the case study using the Mal activity diagrams and SecureUML. Ideally, this work should be done by two distinct teams without being aware of how the other team is modeling with another language. Indeed, in this case , we played the triple role of designer, analyst and judge of the modeling of the case study and the analysis of alignment between Mal activity diagrams and SecureUML with the ISSRM domain model.

Table 7.2: Concept Alignment Between SecureUML and ISSRM Domain Model

Serial No.	Element of the case study	SecureUML	ISSRM domain model
1.	ClientInformation	Class with a stereotype <<ModelElement>> [ClientInformation]	Asset
2.	personalInformation	Attribute of the class with a stereotype <<ModelElement>> [personalInformation]	Business asset
3.	financialStatus	Attribute of the class with a stereotype <<ModelElement>> [financialStatus]	Business asset
4.	BankOfficerPermission	Association class with a stereotype <<Permission>> [BankOfficerPermission]	IS asset
5.	getInformation()	Operations of the class with a stereotype <<ModelElement>> [getInformation()]	IS asset
6.	Attacker	Class with a stereotype <<Role>> [Attacker]	Threat agent
7.	illegalInfoCollection: read	Association class with a stereotype <<Permission>> [illegalInfoCollection: read]	Attack method
8.	BankPhoneNumber	Authorization constraint, constrainedElement [BankPhoneNumber]	Security requirement
9.	Secure ClientInformation system	Whole SecureUML in Fig. 7.6	Control

7.7 Summary

In order to validate our proposed alignments (Table 5.1 and Table 6.1), we have used a case study here. We have identified the key instances of the case study and examine how they are represented by these two languages and to which ISSRM concepts they match. Then we have checked alignment of each element here with the alignment in Table 5.1 and Table 6.1. Our analysis here validates our proposed alignment in Table 5.1 and Table 6.1. Finally, we have discussed the limitations of case study and validation process.

Part IV

V. Conclusion

In this part, we conclude the thesis. We show the limitation and provide a summary of this work. We also discuss answers to the research questions and identify the future work.

CHAPTER 8

Conclusion and Future Work

In this chapter we will summarize our thesis work and discuss about the limitations of this work. We will also provide the conclusion and answer to the research questions. Finally, recommendations for further research conclude the work.

8.1 Limitation

Like any other scientific study, this work also has some limitations. First of all, it is based on theoretical discussion and thus contains a certain degree of subjectivity. One of the limitations is that the work is carried out and validated by only one researcher and revised by another. This is also based on specific scenario. Thus, it might mean that some aspects of the modeling languages (e.g., Mal activity diagrams and SecureUML) or its application could be interpreted and aligned to the ISSRM concepts differently, if the study is performed by other people. Another thing is that the running example also involves the subjective decisions on how to model the problem. For example, we have selected to take the *risk reduction* decision. The *security requirements* could be different if one would take the risk avoidance (or other) decision.

Another shortcoming of this study is that we have focused on specific kind of attack (e.g, spoofing[8]) on the running example and ignored many other attack methods (e.g., Man in the middle attack[9], key logger[10]). Although the example is taken from the literature which reports on a real world scenario but we have not applied it in the practical settings. Thus our analysis remains based on only on the available literature (e.g., Sindre, 2007, Lodderstedt, 2002).

8.2 Answer to Research Questions

The main research problem addressed in this work, as stated in Chapter 1 is "*Support of security modeling language for security risk management at the design level".*

This research problem is divided into three research questions for investigation. Now we will discuss our answer to these research questions.

[8] http://en.wikipedia.org/wiki/Spoofing_attack
[9] https://www.owasp.org/index.php/Man-in-the-middle_attack
[10] http://en.wikipedia.org/wiki/Keystroke_logging#Software-based_keyloggers

RQ 1: *Is there any domain model which would help to understand security risk management at design stage?*

To answer this question we have surveyed and analyzed different security methods (e.g., Braber *et al.,* 2007, Mead *et al.* 2005) and frameworks (e.g., Haley *et al.* 2006, Lee *et al.* 2007). We have resulted in ISSRM domain model (Dubois *et al.*, 2010).

RQ 2: *What are potential security modeling languages at the design stage to support security risk management?*

To answer this question we have investigated the concepts and constructs of different security modeling languages (e.g., Misuse case diagrams, Mal activity diagrams). Some of these language are used at requirement engineering (e.g., Abuse case diagrams, Misuse case diagrams) and some are used at the design stage (e.g., Mal activity diagrams, SecureUMl). This analysis also helps us to understand their concept, constructs and their usage. We have resulted in Mal activity diagrams and SecureUML.

RQ 3: *How could the security modeling language support security risk management at the design stage?*

We have investigated how Mal activity diagrams and SecurUML can support the security risk management. In our analysis, we have found that both these languages can identify and represent *business asset, IS asset, threat agent, attack method, threat, security requirement* and *control* concepts. But they cannot represent *security criterion, risk, event, vulnerability, impact* and *risk treatment* concept.

To finalize our discussion we give the conclusion in the next section.

8.3 Conclusion

Among different security methods (e.g., Braber *et al.,* 2007, Mead *et al.* 2005), we have chosen ISSRM domain model (Dubois *et al.*, 2010) to analyze security modeling language at design stage because of its focus on information system development. The ISSRM domain model is derived and compliant with exiting security standards. It also supports definition of security for the key information system constituents and addresses the information system security risk management process at three conceptual levels: *asset related*, *risk related* and *risk treatment related*. These levels help us to identify and analyze the element of the information system from security risk management perspective. Another major reason to choose the ISSRM domain model is the previous experience (e.g., Mayer, 2009, Matulevičius *et al*, 2008). The domain model has already been used to investigate risk management support in the requirement engineering.

Investigation of existing language alignments (e.g., Mayer, 2009, Matulevičius *et al*, 2008) has helped us to understand how we can analyzes the modeling language at the design stage.

After review of the security modeling languages, we have selected Mal activity diagrams and SecureUML for alignment with the ISSRM domain model. Mal activity diagrams is selected because it can be used as a linkage between late requirement and design stage. We have selected SecureUML because it is defined with an explicit meta model and provides overall

security control mechanism (more specifically role-based access control) for secure information system development.

We have used the background knowledge for the concept alignment between Mal activity diagrams and SecureUML to the ISSRM domain model by using an online banking system example (Annex A.1). Major observations from these alignments (Table 5.1 and Table 6.1) are,

- Neither of these two languages supports the ISSRM domain model in complete extent.
- Mal activity diagrams can only model dynamic security criteria (e.g., integrity of the process) but cannot represent static security criteria (e.g., confidentiality of data).
- SecureUML mainly focuses on static security criteria, but it also supports dynamic security requirements by using authorization constraints.
- Several ISSRM domain model concepts are represented by same constructs (e.g. Activity is used both for *business asset* and *IS asset*). Solution to this could be meta-labeling, tags or even use different colors in the diagram.
- Modeling languages are not able to specify few ISSRM concepts (e.g., *security criterion, vulnerability, risk*). Answer to this problem could be extension of the modeling languages (at the constructs, meta model and semantics level) by introducing additional security and security risk constructs.

We have validated our work by in a case study (Mitnick, 2002) which focuses on personal and financial information management. We have identified the key instances of the case study then used our proposal to model them following the ISSRM process. We have checked whether the concept alignment between construct of modeling language and ISSRM concept focusing on the case study element (Table 7.1 and Table 7.2) matches with our proposed alignment in Table 5.1 and 6.1.

8.4 Future Work

This work has identified some limitations of the modeling language at design stage. The major future work is to implement the identified extension to the modeling languages. Before this implementation could happen, validation should also be done by security experts. Our analysis also opens the way for the interoperability between different modeling languages that are analysed using the same conceptual background, thus, potentially leading to the transformation between different modeling approaches.

Abstract eesti

Turvariskide modelleerimine süsteemi disaini etapis

SecureUML-i vastavusse viimine ISSRM domeenimudeliga

Mohammad Jabed Morshed Chowdhury
Magistritöö

Turvatehnika disain on üks olulisi süsteemiarenduse komponente. Ta peaks läbima tervet süsteemiarendusprotsessi. Kahjuks pööratakse talle paljudel juhtudel tähelepanu ainult süsteemi arendamise ja haldamise ajal.

Paljud turvalise modelleerimise keeled (näiteks *Misuse Case*, *Secure Tropos*) aitavad turvariskejuba nõuete analüüsi etapil hallata. Käesolevas magistritöös vaatleme modelleerimisvahendeid (pahateoskeemid ja SecureUML), mida kasutatakse süsteemi disainil. Täpsemalt, me uurime, kuivõrd need vahendid toetavad infosüsteemide turvariskide haldust (*Information Systems Security Risks Management, ISSRM*).

Töö tulemuseks on tabel, mis seab pahateoskeemid ning SecureUML-keele konstruktsioonid ISSRM domeeni mõistetega omavahel vastavusse. Me põhjendame oma analüüsi ning valideerime saadud tulemusi mitmel illustratiivsel näitel. Me loodame, et saadud tulemused aitavad arendajatel paremini aru saada, kuidas turvariske süsteemi disainietapil arvesse võtta. Peale selle, nende keelte analüüs ühisel kontseptuaalsel taustal annab tulevikus võimaluse neid keeli korraga kasutada ning loodud mudeleid ühest keelest teise teisendada.

Bibliography

Asnar Y., Giorgini P., Massacci F. and Zannone N., "*From Trust to Dependability through Risk Analysis"*. In Proceedings of the 2nd International Conference on Availability, Reliability and Security, IEEE Computer Society Press, 2007.

Asnar Y., Moretti R., Sebastianis M. and Zannone N., "*Risk as Dependability Metrics for the Evaluation of Business Solutions: A Model-driven Approach"*. In Proceedings of the 3rd International Conference on Availability, Reliability and Security, IEEE Computer Society, 2008.

Braber F. D., Hogganvik I., Lund M. S., Stølen K. and Vraalsen F., "*Model-based security analysis in seven steps—a guided tour to the CORAS method*." BT Technology Journal, Volume 25 Issue 1, pages 101–117, 2007.

Bresciani P., Perini A., Giorgini P., Fausto G. and Mylopoulos J., "*TROPOS: an Agent-oriented Software Development Methodology*". Journal of Autonomous Agents and Multi-Agent Systems, Volume 25, pages 203–236, 2004.

Common Criteria for Information Technology Security Evaluation, Version 3.1, 2006. Available at http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R1.pdf

Dubois E., Heymans P., Mayer N. and Matulevičius R., "*A Systematic Approach to Define the Domain of Information System Security Risk Management"*. Book published from Springer-Verlag, 2010. ISBN: 978-3-642-12543-0

Feather M. S., Fickas S., Finkelstein A. and Lamsweerde V. A., "*Requirements and Specification Exemplars*". Journal of Automated Software Engineering, volume 4 issue 4, October 1997.

Firesmith D. G., "*Common Concepts Underlying Safety, Security, and Survivability Engineering"*. Technical Note CMU/SEI-2003-TN-033, Software Engineering Institute, Pittsburgh, Pennsylvania, December 2003.

Giorgini P., Massacci F., Mylopoulos J. and Zannone N., "*Modeling Security Requirements Through Ownership, Permission and Delegation"*. In the proceedings of the 13th Requirements Engineering, pages 167–176, IEEE Computer Society Press, 2005.

Haley C. B., Moffett J. D., Laney R. and Nuseibeh B., "*A Framework for Security Requirements Engineering"*. In Proceedings of the 28th International Conference on Software Engineering, pages 35-42. ACM Press, 2006.

Haley C. B., Moffett J. D., Laney R. and Nuseibeh B., "*Security Requirements Engineering: A Framework for Representation and Analysis"*. IEEE Transactions on Software Engineering, Volume 34 Issue 1, pages 133-153, 2008.

Hoglund G. and McGraw G., "*Exploiting Software: How to Break Code"*. Book published from Addison-Wesley, 2004. ISBN: 0-201-78695-8.

Hoo K. J. S., "*How Much Is Enough? A Risk-Management Approach to Computer Security*". Working Paper, Consortium for Research on Information Security and Policy (CRISP), June 2000.

Hoo K. S., Sudbury A. W. and Jaquith A. R., "*Tangible ROI through Secure Software Engineering".* Secure Business Quarterly, Vo1ume 1, Issue 2, pages 2-7, 2001.

ISO/IEC Guide 73. Risk management Vocabulary Guidelines for use in standards. International Organization for Standardization, Geneva, 2002.

Jackson M., "*Problem Frames: Analyzing and Structuring Software Development Problems".* Book published from ACM Press, 2001. ISBN:0-201-59627-X.

Jürjens J., *" Umlsec: Extending uml for secure systems development".* In proceedings of the 5th International Conference on The Unified Modeling Language, 2002.

Jürjens J., "*Secure Systems Development with UML"*. Book published from Springer Academic Publishers, 2004. ISBN: 3-540-00701-6.

Lamsweerde A. V., "*Elaborating Security Requirements by Construction of Intentional Anti-models"*. In the proceedings of the 26th International Conference on Software Engineering, 2004.

Lee S. W., Gandhi R., Muthurajan D., Yavagal D. and Ahn G. J., "*Building problem domain ontology from security requirements in regulatory documents".* In proceeding of the International Workshop on Software Engineering for Secure Systems, 2006.

Lee S. W., Gandhi R. A., and Wagle S., "*Towards a Requirements-driven Workbench for Supporting Software Certification and Accreditation".* In proceeding of the 3rd International Workshop on Software Engineering for Secure Systems, 2007.

Lodderstedt T, Basin D. and Doser J., "*SecureUML: a UMLbased modeling language for model-driven security".* In Proceedings of the 5th International Conference on the Unified Modeling Language, 2002.

Matulevičius R. and Dumas M., "*A Comparison of SecureUML and UMLsec for Role-based Access Control*". In proceedings of the 9th International Baltic Conference, University of Latvia Press, 2010.

Matulevičius R., Lakk H. and Lepmets M., "*An approach to assess and compare quality of security models*". ComSIS, Volume 8 No 2, Special Issue, May 2011.

Matulevičius R., Mayer N. and Heymans P., "*Alignment of Misuse cases with Security Risk Management".* In Proceedings of the 3rd International Conference on Availability, Reliability and Security, 2008.

Mayer N., "*Model-Based Management of Information System Security Risk*". Doctoral Thesis in Computer Science Namur, Belgium, 2009

McDermott J. and Fox C., "*Using Abuse Case Models for Security Requirements Analysis"*. In Proceedings of the 15th Annual Computer Security Applications Conference, 1999.

Mead N. R., Hough Eric D., and Stehney Theodore R. II. "*Security Quality Requirements Engineering (SQUARE) Methodology"*. Technical Report CMU/SEI-2005-TR-009, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania, 2005.

Mitnick K. "*The Art of Deception: Controlling the Human Element of Security*". Wiley Publishing, Inc., Indianapolis, 2002.

Mouratidis H. and Giorgini P., *"Secure Tropos: A Security-oriented Extension of the. Tropos Methodology"*. International Journal of Software Engineering and Knowledge Engineering, 2005.

Object Management Group (OMG). "*Unified Modeling Language: Superstructure"*. version 2.0, 2004.

Sindre G., "*Mal-Activity Diagrams for Capturing Attacks on Business Processes"*. In proceedings of the Working Conference on Requirements Engineering: Foundation for Software Quality, 2007.

Sindre G. and Opdahl A. L., "*Capturing security requirements by misuse cases"*. In proceedings of the 14th Norwegian informatics conference, 2001.

Turban E., Volonino L., McLean E. and McLean J., "*Information Technology for Management: Transforming Organisations in the Digital Economy*"., The seventh International student edition, 2010. ISBN: 978-0-470-40032-6.

Viega J. and McGraw G., "*Building Secure Software: How to Avoid Security Problems in the Right Way*". Book published from Addison Wesley, 2002. ISBN: 0201-72152-X.

Appendix

A.1 Online Banking

ABC Bank (fictitious) is a renowned bank which has online facilities[11] for its customers. The objectives of online facilities are to improve the quality of service and better deal with the client. The main features of the online banking are transactional, non-transactional, financial administration and transaction approval process. Transition includes deposit and withdraw from the account and payments to third parties, including bill payment and money transfer to other parties. It also includes fund transfer between client's own saving account and transactional account and deposit money for interest. Non transactional functions includes download bank statement and viewing transactions. Online banking system is a server client based system. The main application is hosted in the server and the client access the banking service through any standard web browser (e.g., Mozilla Firefox, Opera). The server includes both database server (e.g., Oracle) and web server (e.g., Apache). The authentication is done using SSL secure connection.

There are few attacks possible against online banking. Among them phishing and pharming are the most popular in hackers community. Cross-site scripting and keylogger/Trojan horses can also be used to steal login information. New attacks are always evolving and getting more and more serious and complex. For example, man in the browser attack, where a Trojan horse permits a remote attacker to modify the client's account number and also the amount in a transaction.

Security experts use several countermeasures to combat against these attacks. Digital certificate is used to stop phishing and pharming. To protect their system from Trojan horses and viruses users/clients should use updated antivirus and anti malware. Users also should be careful when they provide their credentials online. To protect against Man in the Browser attack, user should always use standard browser with latest security patches.

[11] http://en.wikipedia.org/wiki/Online_banking